## Programming Exercise G - CPU Scheduling Algorithms

### Assignment Summary

In this assignment, you will finish a partially-completed program that analyzes different CPU scheduling algorithms. The driver software for the analysis program is already complete. Below is the usage message displayed by the driver.

Usage: a.out <#processes> <time quantum> -user

Usage: a.out <#processes> <time quantum> -random <max runtime>

<#processes> : Positive integer  
<time quantum> : Positive integer  
<max runtime> : Positive integer

The choice of "-user" tells the program to prompt the user and read in each process running time from the keyboard. The choice of "-random" tells the program to randomly generate the running times with <max runtime> as the maximum running time.

The program stores all process information in an array of records that is passed to each scheduling algorithm. Information on the format of the record and the use of each field are described in later sections of this document.

Your assignment is to finish this program by implementing a simulation of the First Come, First Served (FCFS), non-preemptive Shortest Job First (SJF), and Round Robin scheduling algorithms (as demonstrated in class), and printing the analysis results of those simulations. All results shall be sent to standard out and use the buffered I/O functions (e.g, printf()). All times are assumed to be in milliseconds.

Your analysis shall show the results of running the entire set of processes in a time-sharing approach on one CPU using each of the scheduling algorithms. The analysis shall assume that each process initially arrives in the ready queue at the same time as the other processes. It shall also assume that after a process is given the CPU to use, the process gives up the CPU only after its allotted time ends.

The analysis results shall consist of a table of time data for each algorithm and then a summary of that data as shown in the **sample-run.txt** file. (Note that the driver module prints the text denoting the start of each table.) The table of time data shall show the time usage for each process and shall be formatted as shown in the sample run file. The summary information printed below the table shall show the average waiting time and the average turn-around time for that scheduling algorithm. The average waiting time is the average of the **total amount of time** that each process spends waiting in the ready queue. The average turn-around time is the average of the **final** completion time for each of the processes.

Note that for the round robin algorithm, each process shall run in sequence for its allotted quantum time starting with process zero. Also note that for the round robin report, the summary information shall also show the time quantum that was used (i.e., that was provided by the user on the command line).

### Process Record Type

typedef struct  
   {  
   int processNbr; // The specific process number (i.e., 1) for a process  
   int totalTime; // Total time that the process needs to run  
   int remainingTime; // Remaining time that the process needs to run  
   int stopTime; // Most recent time when the process stopped running  
   int waitSum; // Sum of the time that the process has waited to run  
   int isFinished; // Indicates if the process has completely finished running yet  
   } processDataType;

### Intialization and Use of Record Fields in the Program

processDataType processes[MAX\_PROCESSES];  
. . .  
. . .  
. . .  
processes[i].processNbr = i; // "Read only"; used by FCFS, SJF, and Round Robin  
processes[i].totalTime = jobTime; // "Read only"; used by FCFS, SJF, and Round Robin  
processes[i].remainingTime = jobTime; // "Read and write"; used only by Round Robin  
processes[i].stopTime = 0; // "Read and write"; used only by Round Robin  
processes[i].waitSum = 0; // "Read and write"; used only by Round Robin  
processes[i].isFinished = FALSE; // "Read and write"; used only by Round Robin

### Assignment Directions

1. Download the following two source code files from the Assignments page of the COSC 3503 course website:
   * **program-driver.c** - This source code file contains the code to gather the running time information requested for each of the processes, store the information in an array of records, and pass the array to each of the scheduling algorithm functions as they are called. Make no changes in this file.
   * **scheduling-algorithms.h** - This source code header file contains #define constants, a type definition, and function prototypes for the scheduling algorithm functions. Make no changes in this file.
2. Create a source code file named "scheduling-algorithms-module.c". At the start of the file place a #include directive for the scheduling-algorithms.h file. Next, type in your implementation of the FCFS, SJF, and Round Robin scheduling algorithm functions whose prototypes are supplied in the scheduling-algorithms.h file. Follow the same algorithm approach demonstrated in class. Notice the additional fields provided in the processDataType record. Use these fields as needed to help you in implementing each scheduling algorithm. You may add one or more helper functions to this file as needed to assist you in implementing the FCFS, SJF, and Round Robin algorithms; however, use no global variables and make no changes to the contents of the scheduling-algorithms.h file.
3. Compile and link the analysis driver file and your scheduling algorithms implementation file into a single executable file. Run the program using a variety of values for the number of processes, the time quantum, the maximum running time, and the individual process running times to see the results produced by each scheduling algorithm. Note especially the differences in the **average waiting time** and **the average turn-around time** produced by each scheduling algorithm.
4. Only submit your scheduling-algorithms-module.c file to Blackboard. Do **not** submit the other two source code files because the instructor already has them.

### Implementation Constraints

* Follow the same coding standards given in previous assignments
* Use no tab characters in your program output