Operating Systems
Assignment 3

Objectives:

Evaluating the relative merits of CPU scheduling algorithms.

Task: CPU Scheduling - Part II.

For this assignment, we will re-use the code we built in Assignment 1, but modify it to perform different algorithms, and collect performance statistics.

Rather than using real time - which can be inconsistent (due to other background processes, etc.), we will simply use a counter to emulate elapsed time. The counter will start at zero, and count up in units of one until all processes have completed.

What you need to do:

Modify your CPU emulator to 'run' the processes by decrementing the remaining run time of the process.*
Modify your CPU to accept time-quantum arguments on startup. You can tell it how long to run before stopping and sending the process back to the scheduler, or tell it to run a process until completion.*

Modify your scheduler-emulator to keep track of wait times, and total elapsed time.
You will need to be able to record the total waiting time per each process.
Modify your scheduler to add two units of time for every context switch.
Modify your scheduler to add items to the ready queue at their arrival time.

Run the list of processes using the following algorithms:

FCFS - no pre-emption
SJN - no pre-emption
Priority - no pre-emption
RR - FCFS with time quanta of 10, 100, and 1000 units.
RR - SRTN with 50-unit time quantum
RR - Priority with 50-unit time quantum

For each algorithm, record and report the following information:

Average Turnaround time (time from submission to completion).
Average wait time
Longest wait time
Total run time to complete all jobs (sum of all processes' run time + their context-switch time).
Total number of context switches

Answer the following questions about the algorithms:

Which have the longest / shortest total run time?
Which are the most responsive / least responsive?
Which have the longest / shortest average wait time?
Which has the longest total wait time for any single process? Shortest? Why?
Which have the best / worst CPU utilization (think of time lost to context switches).

Extra Credit:

Make a plot to compare the behavior of these algorithms
What to plot is up to you - average wait times, longest wait time, etc.
Be sure to clearly label axes, units, titles, etc.
You must use a programming language to make this plot - don't turn in MS Excel =\

Extra Credit II:

Implement Priority scheduling with pre-emption, using two processes and IPC.
You'll need to be able to send an "interrupt" message to stop the current process from running, and load a new one if a higher-priority process enters the ready queue.

Notes:

* You do NOT have to send processes back and forth between your two processes if you don't want.
It shouldn't be hard, but you may experience *blocking*, which will mess up your statistics-keeping (if you used sockets / blocking calls).
You may handle the CPU emulation all on one side if you wish - for simplicity.
(this assignment is more about algorithms than the intricacies of inter-process communication).

You may want to store the total wait time in the process' PCB - or keep a separate data structure for statistics - it's up to you.

To keep track of statistics across runs of different algorithms, you may want to append your time observations to a file, then compute statistics from that file later - after all algorithms have run.

It is recommended that you build your program to run different algorithms based on command-line input arguments - so that you can quickly re-run algorithms, rather than modify your underlying code to change algorithms.

What to turn in:

1. A (very brief) summary PDF, showing a table of your recorded timing results, and your answers to the questions above.

2. Your source code - complete enough that it can be executed and tested.

3. Any compile / make / run instructions you used, so your code can be executed and tested as you intended.
This can be a make file, shell script, or a simple series of instructions in a text file - just include whatever you did to compile & run your code.

Zip files and other compressed folders will receive zero credit. Just attach your files to the email.