EECS3221 Project

You have to work individually. You are not allowed to view or exchange documents with your peers. We treat all sorts of cheating very seriously. Do not copy code from anywhere, even as a "template". No late submission will be accepted. All assignment work must be done on the EECS `red` *server. Your work will be graded based on: i) correctness of programming logic; ii) clarity of your code and your coding style; iii) whether your code follows the specifications. It is your responsibility to explain clearly every detail you do in the code with appropriate comments to avoid any possible confusion in marking.*

Multiple-processor CPU Scheduling Simulation

The CPU Scheduler is a key component of any multiprogramming operating system (OS) kernel. It is important to understand the details of the CPU scheduling in order to fully comprehend the dynamic behavior of the operating systems.

In this project, you are asked to write some C programs to simulate the following three CPU scheduling algorithms for a multi-core computing system consisting of four (4) homogeneous CPUs:

1) FCFS (First-Come-First-Served) scheduling.
2) RR (Round Robin) scheduling with time quantum q=3 milliseconds, q=10 milliseconds, and q=45 milliseconds, respectively.
3) Three-level feedback-queue (FBQ) preemptive scheduling employing queue Q0 (RR, quantum q=8 milliseconds), queue Q1 (RR, quantum q=20 milliseconds), and queue Q2 (FCFS) as shown in Figure 5.9 in the textbook. Read the corresponding text for details about this scheduling algorithm and include in your implementation the following:
   (a) a process exceeding its time quantum should be preempted and demoted by placing it into Q1 if its original queue was Q0 or into Q2 if its original queue was Q1
   (b) a process finishing its I/O bursts should be promoted by placing it into Q0
   (c) a process preempted for servicing a higher priority queue should be placed in the beginning of its original ready queue

When you implement your simulations, if two or more processes are identical in terms of scheduling criterion, you should give precedence to the process that has the smallest process ID (PID) number.

Based on the given static CPU workload in the `CPULoad.dat` text file your programs must perform the necessary simulations and answer all the following questions for each of the above CPU schedulers:

1) What is the average waiting time?
2) What is the average turnaround time?
3) When does the CPU finish all these processes? What is the average CPU utilization by this time point? (Note that at any time instance, CPU utilization is 400% if all 4 CPUs are running, 300% if only 3 CPUs are running, 200% for 2 CPUs, 100% for only 1 CPU, and 0% if no CPU is running.)
4) How many context switches occur in total during the execution? (Note that you should count as context switches only the cases where a process is preempted during its CPU burst, and not the cases where a process terminates or just finishes its current CPU bursts and goes to I/O.)
5) Which process is the last one to finish?

In this project, you need to write three C programs to implement simulations with the above CPU schedulers. For the FCFS scheduler, put your program in a file named `fcfs.c`, for the RR scheduler in a file named `rr.c,` and for the FBQ scheduler in a file named `fbq.c`. When the programs run, they should print the results to the standard output (`stdout`).

For this project, you are provided with two helper files (`sch-helpers.c` and `sch-helpers.h`), which include C functions for loading data from a CPU load file, for sorting processes, and for linked-list based scheduling queue operations. Please read carefully the comments in the files for better understanding of the helper functions. You are free to modify the contents of those two files to fit your needs but you are not allowed to change the file names. You must submit those files as part of your solution even if you make no content changes.

Data Format

The CPU workload data file `CPULoad.dat` contains textual data in the following format (all times are in millisecond):

```
PID Arrival_time CPU_burst#1 (IO_burst#1) CPU_burst#2 (IO_burst#2) ...
```

e.g.

```
0   04 (100) 12 (67) 2 ...
1   13 7 (210) 20 (23) 67 ...
```

The content of the `CPULoad.dat` file is provided as an example and your programs should not be limited to it; they should work with any file content as long as it is properly formatted as explained above.

Submission of Project Part I (P1.1)

You should submit the file with your C code for the FCFS scheduler (`fcfs.c`) the two helper files (`sch-helpers.c` and `sch-helpers.h`), and your report (`reportP1.1.pdf`) before the first deadline as shown on the course eClass site.

Your `fcfs.c` program should compile and run on the `red` server as follows:

```
gcc -Wall -o fcfs fcfs.c sch-helpers.c
./fcfs < CPULoad.dat
```

Make sure that you include the following information (please complete) as a comment in the beginning of all your C programs:

/

Family Name:

Given Name:

Student Number:

EECS Login ID (the one you use to access the `red` server):

YorkU email address (the one that appears in eClass):

**/*

Your report (1-2 pages in PDF format) should be structured as follows:

1) A general outline of your understanding of the assigned work.
2) A clear statement about the assigned work/components you believe you have done/completed successfully.
3) A statement about the work you believe you might have not completed successfully (feel free to comment on related problems, if any).
4) Anything else related to your work that you might wish to comment upon.

Please copy the above entries and paste into your report to use as a template when typing.

After the first deadline, a sample simulation code for an FCFS scheduler will be provided to help you debug your programs and improve your C coding.

Submission of Project Part II (P1.2)

You should submit the files with your C code for the RR and FBQ schedulers (`rr.c` and `fbq.c`), the two helper files (`sch-helpers.c` and `sch-helpers.h`) and your report (`reportP1.2.pdf`) before the second deadline as shown on the course eClass site.

Your `rr.c` and `fbq.c` programs should compile and run on the `red` server as follows:

```
gcc -Wall -o rr rr.c sch-helpers.c
./rr 3  < CPULoad.dat
./rr 10  < CPULoad.dat
./rr 45  < CPULoad.dat

gcc -Wall -o fbq fbq.c sch-helpers.c
./fbq 8 20  < CPULoad.dat
```

Make sure that you include the following information (please complete) as a comment in the beginning of all your C programs:

/

Family Name:

Given Name:

Student Number:

EECS Login ID (the one you use to access the `red` server):

YorkU email address (the one that appears in eClass):

**/*

Your report (1-2 pages in PDF format) should be structured as follows:

1) A general outline of your understanding of the assigned work.
2) A clear statement about the assigned work/components you believe you have done/completed successfully.
3) A statement about the work you believe you might have not completed successfully (feel free to comment on related problems, if any).
4) Anything else related to your work that you might wish to comment upon.

Please copy the above entries and paste into your report to use as a template when typing.