EECS 3221 (Fall 2018) Project

You need to work individually for this project. 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". As part of marking procedure, TA's always test your codes using a popular tool to detect any software plagiarism. No late submission will be accepted. You are required to program in your PRISM account. 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 specification. 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

A CPU Scheduler is a key component of any multiprogramming operating system (OS) kernel. It is very important to understand all details about the CPU scheduler if you want to know dynamic behaviors of an operating system.

What to do?

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 CPU's:

(a) FCFS (first-coming-first-serving) scheduling.

(b) RR (round robin) scheduling with time quantum q=2 milliseconds, q=12 milliseconds, q=50 milliseconds, respectively.

(c) Three-level feedback-queue (FBQ) preemptive scheduling with q1=10 milliseconds and q2=30 milliseconds, as shown in Figure 6.7 in the textbook. Read the corresponding text for details about this scheduling algorithm.

Based on a given static CPU workload (download here), your program must calculate 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 average CPU utilization by this time point? (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, 0% if no CPU is running.)
4. How many context switches occur in total during the execution? (**How to count context switch for this question**: you should count as context switch only for those cases where a process is preempted during its CPU bursts, not for those 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 the above CPU schedulers. For FCFS scheduler, your program is called "`fcfs.c`". For RR scheduler, your program is called "`rr.c`". For FBQ scheduler, your program is called "`fbq.c`". When the programs run, they should print out the answers to all of the above questions to the standard output.

In this project, you are provided with some helper functions, which includes C functions to load data from a CPU load file, to sort processes, to do linked-list based scheduling queue implementation. You can download them (sch-helpers.c and sch-helpers.h) and directly use them in your code. Please read closely the hints at the beginning of `sch-helpers.c` for how to use these helper functions. You are free to make any modifications to these two files to fit your need.

Data Format

The CPU workload data file can be downloaded from the course Web, which is an ASCII file made in Unix system. Each line represents one process with the following format: (all numbers are in unit of millisecond)

pid arrival_time 1st_CPU_burst (1st_IO_burst) 2nd_CPU_burst (2nd_IO_burst) ...
e.g.

```
0  0  4 (100) 12 (67) 2 ...
1  13 7 (210) 20 (23) 67 ...
```

where each I/O burst includes both the time waiting in the device queue and the time spent in actual I/O operations. **When you implement each scheduler, if two or more processes are identical in terms of scheduling criterion, you should give priority to a process that has the lowest *pid* number.**

What to submit?

Part I (40%): You should submit the first code `fcfs.c` as well as the above two helper files before the first deadline as specified in the course website. Run the following commands to submit:

```
   submit 3221 project1 fcfs.c
   submit 3221 project1 sch-helpers.c
   submit 3221 project1 sch-helpers.h
```

Your program should compile in your PRISM account as follows:

```
  gcc –Wall –o fcfs  fcfs.c sch-helpers.c
```

And your code should run like:

```
  fcfs  < CPULoad.dat
```

After the first deadline, a sample code of FCFS scheduler will be posted online for you to debug your code and learn how to write a good C program. Then you will continue to finish the remaining schedulers by the second deadline.

Part II (60%): You should submit the rest two schedulers `rr.c` and `fbq.c` before the second deadline as specified in the course website. Run the following commands to submit:

```
   submit 3221 project2 rr.c
   submit 3221 project2 fbq.c
   submit 3221 project2 sch-helpers.c
   submit 3221 project2 sch-helpers.h
```

Both `rr.c` and `fbq.c` should compile and run in the same way as `fcfs.c`.

```
gcc –Wall –o rr  rr.c sch-helpers.c

gcc –Wall –o fbq  fbq.c sch-helpers.c
```

And `rr` and `fbq` should take a time slice as command-line argument and run as follows:

```
rr 2  < CPULoad.dat

rr 12  < CPULoad.dat

rr 50  < CPULoad.dat

fbq 10 30  < CPULoad.dat
```

At the beginning of each of your C files, include the following information (please complete) for yourself:

# *Family Name:*

# *Given Name:*

# *Student Number:*

# *CSE Login:*

No hardcopy is needed to submit for this project.