More information about the CS201 graduate assignment

Here are the kinds of events that can happen in the system:

1. PROCESS_SUBMITTED: when a process enters the system
2. PROCESS_DISPATCHED: when a process moves running state, on an unoccupied CPU
3. PROCESS_TERMINATED: when a process leaves the system, because it has achieved its total runtime
4. TIME_SLICE_EXPIRED: when the time that a process has been in running state exceeds the time quantum
5. IO_REQUEST: when a process in running state gets an I/O request
6. IO_COMPLETE: when the I/O for a process finishes, and it moves back to the ready state

Every event will have an entry in the event queue. It makes sense for each entry in the event queue to have this information

* time (= priority)
* event type
* process

Each process should have a unique (integer) process id.

Here's how a start the simulation
* create a new process at time = 0 and put it the queue, with eventType = PROCESS_SUBMITTED; assign to this process the total run time, a burst time (the time between I/O faults) and I/O service time

Then, start your while loop:

```
event = eventQueue.dequeue();
while ( ! stop && event != null) {
  systemClock.update(event.getTime());
  event.handleEvent();
  event = eventQueue.dequeue();
}
```

The stop variable will become true if a stop time was provided, and the systemTime > stopTime.

The handleEvent() will do different things depending on the event type.

(1) PROCESS_SUBMITTED:
(1a) if the ready queue is empty
  - assign the process to a CPU
  - put a new event in the event queue
    - if the burstTime is longer than the quantum, then the new event is "TIME_SLICE_EXPIRED", with time = currentTime + quantum
    - else put a new event "IO_REQUEST" in the event queue, with time = currentTime + burstTime
    - else if the timeRemaining < quantum, then put a PROCESS_TERMINATED event in the eventQueue with time = currentTime + timeRemaining
(1b) else enqueue the process in the ready queue
(1c) in both cases, create a new process and put a new event PROCESS_SUBMITTED in the eventQueue at currentTime + r, where r is a random number representing the interprocess arrival time

(2) TIME_SLICE_EXPIRED
* update the timeRemaining for this process: subtract the quantum
* enqueue the process in the ready queue
* dequeue the next process from the ready queue and assign it to this CPU, doing the same actions shown in (1a) above

(3) IO_REQUEST
* update the timeRemaining for this process: subtract the burst time
* put a new event IO_COMPLETE in the eventQueue, with time = currentTime + the IO_serviceTime for this process
* dequeue the next process from the ready queue and assign it to this CPU, doing the same actions shown in (1a) above

and so forth.

Each time you dequeue an event from the event queue, update the systemTime to be the time value of that event.

In this way, everything that the system does will correspond to an event in the eventQueue, and the system will do only things that correspond to events in the eventQueue