

CS 201 OPERATING SYSTEMS (FALL 2005)

ASSIGNMENT 3 (Due on Nov 3)

- 1 Do the following exercises from the textbook: 7.3, 7.4, 7.5, 7.8, 7.11, 7.16.
- 2 Write a program(s) which simulates the events in a multiprogrammed computer system according to the following guidelines:
 - Round-Robin scheduling is used. Time Slice = 5.
 - You may assume that the number of processes does not exceed 10.
 - Use a counter to implement a logical clock.
 - At each increment of the counter your program should check if the current counter value (time) matches with an arrival time of any process, and enter arriving process(es) automatically into the tail of the ready queue.
 - If a process finishes completely then it leaves the system, and the process at the head of the ready queue is allocated the CPU.
 - If time slice is expired and the process does not finish then the process is inserted into the tail of the queue, and a process (possibly the one just inserted) at the head of the queue is allocated the CPU. Note that the system needs to keep track of remaining burst times. Have a Process Control Block (PCB) for each process to store this information (remaining burst time).
 - Simulation will continue until all processes finish completely.
 - An input file includes information for all processes, their arrival times, and a CPU-burst. Input lines are of four possible formats (Assume that there is exactly one line of input for CPU burst):

```
BEGIN      <process number (<=10)>  <arrival time>
CPU        <CPU burst>
END /* Process description ends */
```

The following is an example:

```
BEGIN      1      0
CPU        7
END

BEGIN      2      5
CPU        4
END
```

- You should not change the input file format as we will test your program with our input files.
- You may assume that the input is correct.

Your program should report which process executes as there occurs a change, or at the beginning of every time slice. For example see the output below for the input given above.

- Report the events in the following format:

```
TIME: 0   PRO 1 EXEC
TIME: 5   PRO 2 EXEC
TIME: 9   PRO 1 EXEC
```

- 3 Write a program(s) which simulates the events in a multiprogrammed computer system according to the following guidelines:

- Semaphores are used as synchronization primitives. There are at most 10 semaphores defined. The wait and signal operations on the semaphores are atomic.
- You may assume that the number of processes does not exceed 10.
- Scheduling is non-preemptive First-Come-First-Served (FCFS). A process continues to execute until a wait operation causes it to sleep on a semaphore.
- Use semaphore implementation which associates a list of waiting processes for each semaphore (See page 203 of the textbook).
- Use a counter to implement a logical clock.
- At each increment of the counter your program should check if the current counter value (time) matches with an arrival time of any process, and enter arriving process(es) automatically into the tail of the ready queue.
- If a process finishes completely then it leaves the system, and the process at the head of the ready queue is allocated the CPU.
- A process issuing a wait on a semaphore is blocked, and a process at the head of the ready queue is allocated the CPU. Note the system needs to remember the next input line in the process definition as the point to resume when the process is woken up at a later time. Have a Process Control Block (PCB) for each process to store a pointer (or index) to process definition (file or array) in your implementation.
- Simulation will continue until all processes finish completely.

- WE DEFINE AND INITIALIZE SEMAPHORES BY A STATEMENT OF THE FOLLOWING FORMAT:

```
SEM          <semaphore number>      <initial value>
```

- An input file includes information for all processes, their arrival times, CPU-bursts, wait and signal on semaphores. Input lines are of six possible formats:

```
SEM          <semaphore number>      <initial value>
BEGIN        <process number (<=10)>  <arrival time>
CPU          <CPU burst>
WAIT         <semaphore number>
SIGNAL       <semaphore number>
END /* Process description ends */
```

The following is an example:

```
SEM          1      0

BEGIN        1      0
WAIT         1
CPU          7
END

BEGIN        2      5
CPU          4
SIGNAL       1
END
```

- You should not change the input file format as we will test your program with our input files.
- You may assume that the input is correct.

Your program should report which process executes as there occurs a change. For example see the output below for the input given above.

- Report the events in the following format:

```
TIME: 5   PRO 2 EXEC
TIME: 9   PRO 1 EXEC
```