Atal Bihari Vajpayee Indian Institute of Information Technology & Management

Minor Examination

Sub: Operating Systems **Course Code: CS207**

Time: 2 Hours **Date: 26/Feb/2025** **Max. Marks: 30**

Note: Attempt all questions and answer them appropriately.

No doubts will be cleared during examination.

1. A computer can operate under two operating systems, OS1 and OS2. A program P always executes successfully under OS1. When executed under OS2, it is sometimes aborted with the error "insufficient resources to continue execution," but executes successfully at other times. What is the reason for this behavior of program P? Can it be cured? If so, explain how, and describe its consequences. (Hint: Think of resource management policies.) [4]

2. Explain the 7-state transition for a process with example. [4]

3. Can deadlocks arise in the following situations? Provide the justification. [4]
   a. A system performs partitioned allocation of resources to programs.
   b. A set of programs communicate through message passing during their execution.

4. Consider N processes sharing a single CPU in a round-robin fashion (N ⩾2). Assume that each context switch takes S ms and that each time quantum is Q ms. For simplicity, assume that processes never block on any event. Find the maximum value of Q (as a function of N, S, and T) such that no process will ever wait in the ready queue for more than T ms. [4]

5. Explain Dead lock detection Banker's Algorithm and safety algorithm? Consider the following snapshot of a system: [8]

Process	Allocation	Max	Available
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

Answer the following questions using the Banker's algorithm:

a. What will be the contents of matrix need?
b. Draw the safe sequence?
c. If a request from process P1 arrives for (0,4,2,0) can the request be granted immediately?

6. Consider a multi-processor system with three processors CPU1, CPU2, and CPU3. All the three processors share a common ready queue, and each processor internally runs its own scheduling algorithm. Whenever appropriate (like a processor is free, or it is running a low- priority process), the internal scheduler runs to schedule the next process from the common ready queue. As the internal scheduling algorithm, CPU2 and CPU3 use round-robin scheduling with time quanta 3ms and 2ms, respectively, whereas CPU1 uses the pre-emptive priority scheduling algorithm. CPU1 considers the common ready queue as a priority queue, whereas CPU2 and CPU3 considers this queue as a FIFO queue. If CPU1 runs a low-priority process, and a high-priority process arrives, the priority scheduler of CPU1 suspends the currently running process and schedules the high-priority process (even if the other processors are free). If multiple processors are free, the free processor with the lowest index gets the opportunity to run its internal scheduler and schedules the next process. Assume that the round-robin scheduler of CPU2 or CPU3 always inserts a pre-empted process (when its time quantum is over) to the ready queue (even if the queue was empty before the insertion). Note that one process may run on multiple processors during its lifetime.

Consider four processes arriving to the system at the arrival times and with priorities as specified in the following table (lower priority number means higher priority, so P4 is the highest-priority process, and P1 is the lowest-priority process). Assume that each process requires two CPU bursts and two I/O bursts for completion (with the irrespective durations as specified in the table). All times are in ms.

Process	Arrival Time	Priority	First CPU burst	First I/O burst	Second CPU burst	Second I/O burst
P1	0	4	4	2	4	3
P2	2	2	10	2	3	2
P3	3	3	8	4	8	4
P4	7	1 (H)	3	1	3	1

Draw the Gantt charts for the three processors. In each Gantt chart, clearly mark (alongside the CPU usage by the processes) the time instances showing (i) when a process starts and finishes its first I/O, (ii) when a process starts and finishes its second I/O, (iii)when a process terminates (after its second I/O), and (iv) the durations when the CPU remains idle. [6]