IIT Guwahati - Department of Computer Science & Engineering

CS343-Operating Systems- Mid Semester Exam [22.09.2021]

Total 30 marks, 90 minutes.

SOLUTION SET

Section I: Essay Type (3 questions, 4 marks each). Write answers with necessary explanation in the given space.

Question 1:
Two shared resources R_1 and R_2 are used by processes P_1 and P_2. Each process has a certain priority for accessing each resource. Let T_{ij} denote the priority of P_i for accessing R_j. A process P_i can snatch a resource R_k from process P_j if T_{ik} is greater than T_{jk}.
Given the following conditions:

1. $T_{11} > T_{21}$
2. $T_{12} > T_{22}$
3. $T_{11} < T_{21}$
4. $T_{12} < T_{22}$

Explain which all combinations of the above conditions (1 combination is 2 conditions) ensure that P_1 and P_2 can never enter in deadlock?

Solution:
If all the resources are allocated to one process, then deadlock will never occur. So, if we allocate both R1 and R2 to process P1 or both R1 and R2 to process P2 then deadlock can be prevented. When one process completes its execution then both the resources are allocated to the other process. So, two combinations will ensure deadlock freedom. **[1 mark for explanation]**
Combination 1: conditions 1 and 2. **[1.5 marks]**
Combination 2: conditions 3 and 4. **[1.5 marks]**

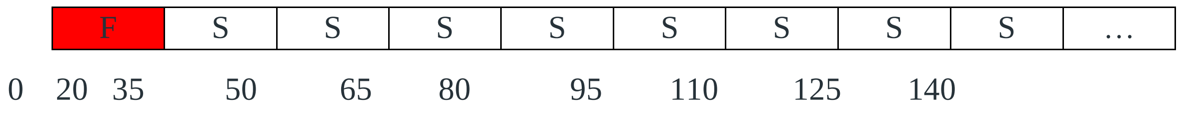
Question 2:
Consider an automated war fare equipment that is fitted on a drone. The system does two operations defined by the following processes:
1. Fire() – [CPU burst time = 60 seconds] Analyse recent images captured by Scan() and identify missiles that are approaching to the drone from the enemy camp and shoot a counter missile to crash the incoming missile.
2. Scan() – [CPU burst time = 15 seconds] Initiate capture of high resolution images of the terrain under the drone and store the captured images for the Fire () to analyse further and take action.
The Fire() and Scan() are triggered in every 100 seconds and 15 seconds, respectively. Here triggering means the process is in ready state. Assume the first triggering of Fire() and Scan () occurs at T=0 and T=20, respectively. The design permits only one process to be in the running state based on pre-emptive shortest job first scheduling.
(a) Comment on the efficiency of the system in meeting its objectives.
(b) Keeping both CPU burst time of the processes and CPU scheduling algorithm unmodified, suggest your recommendations to improve the efficiency of the system.

Solution:

(a) As per the specification given above, Fire() will get triggered @ T=0, 100, 200, 300, etc.. and each instance of Fire() has a CPU burst of 60 seconds.

Similarly, Scan() will get triggered @ T=20, 35, 50, 65, 80, 95, 110 etc.. and each instance of Scan() has a CPU burst of 15 seconds.

At T=0, Fire() will run as there is no other process in ready state. As per pre-emptive shortest job first scheduling, Scan () is always shortest job than Fire(). At T=20 it will pre-empt Fire(). Thereafter never Fire () can be in running as always a new instance of Scan()) is ready when current Scan() completes execution. If Fire() is not running the mission will fail as it fails to short down missiles from enemy camp. **[2 marks]**



(b) The system is efficient only if new instances of Fire() and Scan() completes execution in a defined time window and Fire() and Scan () are not starving to get into execution. For this to happen, exactly one instance of Fire () and at least 1 & at most 2 instances of Scan() should execute fully in a time window of 100 seconds in a repetitive pattern. **[2 marks]**
We have to ensure that one Scan() and minimum one Fire() to be executing fully within 100 seconds. Invoking of additional instances of Fire() is ok provided there should not be starvation for Fire () process.
Example 1: Scan () at T=0, 20, 100, 120, 200, 220.. and Fire () at T=40, 140, 240..
Example 2: Scan () at T=0, 80, 100, 180, 200, 280.. and Fire () at T=20, 120, 220..
Example 3: Scan () at T=60, 80, 160, 180, 260, 280.. and Fire () at T=0, 100, 200..

Question 3:

Let R be a reader process and W be a writer process. Let F be the implementation of first readers writers problem and S be the implementation of second readers writers problem.

(a) List out the scenarios where a new R is made to wait in F.
(b) List out the scenarios where a new R is made to wait in S.
(c) List out the scenarios where a new W is made to wait in F.
(d) List out the scenarios where a new W is made to wait in S.

Solution:

In the first readers-writers problem, no readers will be kept waiting unless a writer has already obtained permission to used the shared resource. In the second readers-writers problem, new readers will be kept waiting when there is at least one writer that is in waiting state or one writer is accessing the resource. This is accomplished by forcing every reader to lock and release the semaphore individually. The writers on the other hand don't need to lock it individually. Only the first writer will lock the semaphore and then all subsequent writers can simply use the resource as it gets freed by the previous writer. The very last writer must release the semaphore, thus opening the gate for readers to try reading.

(a) List out the scenarios where a new R is made to wait in F.
When a W is already using the shared resource. **[1 mark]**
(b) List out the scenarios where a new R is made to wait in S.
When a W is already using the shared resource OR at least one W is in waiting. **[1 mark]**
(c) List out the scenarios where a new W is made to wait in F.
When at least one R/W using the shared resource. **[1 mark]**
(d) List out the scenarios where a new W is made to wait in S.
When at least one R/W using the shared resource. **[1 mark]**

Section II: File upload Type (3 questions, 6 marks each). Write answers with necessary explanations in a paper and upload the scan copy of the page(s). Maximum 2 pages per questions.

Question 4:

Consider an operating system that uses dynamic priority based pre-emptive scheduling. There are 4 user processes P1, P2, P3 and P4 with arrival time T= 4, 8, 12, and 52, respectively. The CPU burst of these processes are 12, 20, 16, and 12 cycles, respectively and they all have a priority value 2 at their arrival time. The process is eligible for considering for running state only if its priority value is 0. The priority value of a process will be decremented by 1, if it is in waiting state for 4 consecutive cycles. Priority values of a waiting state process will not go to negative and will stay at 0 till it is promoted to running state. Similarly, a process that is in running state for 4 consecutive cycles will get its priority value incremented by 1 thereby moving to waiting state again. If any tie occurs for picking a process for running state, then the winner is that process with least balance CPU burst time and still if the tie is not resolved, then the winner is that process that has the earliest arrival time to the system. When no user process is eligible to be in the running state, OS process called watcher will be running.

(a) Draw a neat labelled Gantt chart of the scheduling of these processes.
(b) How long process P3 will take to complete after the completion of process P2?
(c) What is the turnaround time for process P4?
(d) How many cycles the watcher will run between T=0 and T=N, where N is the time when the last process completes its CPU burst?

Solution:

First, we observe the time at multiples of 4 (where pre-emption and priority change happens). Each process entry is represented as A | B. Where A is the remaining CPU Burst Time and B is the current priority value. **[Illustration of the following time table is not mandatory]**

Time (T)	P1	P2	P3	P4
0	-	-	-	-
4	12 | 2	-	-	-
8	12 | 1	20 | 2	-	-
12	**12 | 0**	20 | 1	16 | 2	-
16	8 | 1	**20 | 0**	16 | 1	-
20	**8 | 0**	16 | 1	16 | 0	-
24	4 | 1	**16 | 0**	16 | 0	-
28	**4 | 0**	12 | 1	16 | 0	-
32	0 | -	**12 | 0**	16 | 0	-
36	-	8 | 1	**16 | 0**	-
40	-	8 | 0	12 | 1	-
44	-	4 | 1	**12 | 0**	-
48	-	**4 | 0**	8 | 1	-
52	-	0 | -	**8 | 0**	12 | 2
56	-	-	4 | 1	12 | 1
60	-	-	**4 | 0**	12 | 0
64	-	-	0 | -	**12 | 0**
68	-	-	-	8 | 1
72	-	-	-	**8 | 0**
76	-	-	-	4 | 1
80	-	-	-	**4 | 0**
84	-	-	-	0 | -

Number of time process run:
W = 6 X 4 = 24
P1 = 3 X 4 = 12
P2 = 5 X 4 = 20
P3 = 4 X 4 = 16
P4 = 3 X 4 = 12
TOTAL : 84

(a) Gantt chart according to dynamic priority and constraints given for scheduling a process to CPU. In the Gantt chart, we have to show break point after every 4 cycles we have to consider the priorities, remaining burst time and change them accordingly as shown in above table. **[3 marks for correct labelled Gantt chart from T=0 to T=84]**

W	W	W	P1	P2	P1	P2	P1	P2	P3	P2	P3	P2	P3

0 4 8 12 16 20 24 28 32 36 40 44 48 52 56

W	P3	P4	W	P4	W	P4

56 60 64 68 72 76 80 84

By using Gantt chart we can clearly observe that all process will be finished at T=84 assuming T=0 as the starting time.

(b) Process P3 completes @ T=64 and P2 completes @ T=52
Hence P3 takes additional 12 cycles to complete after P2. **[1 mark]**

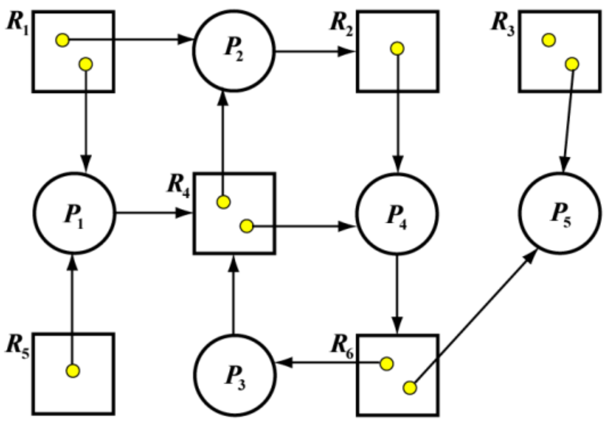
(c) Turnaround time (P4) = Completion time – Arrival time = 84 - 52 = 32 cycles **[1 mark]**

(d) Watcher will be running for 24 cycles. **[1 mark]**

Question 5:

Consider the following resource allocation graph.

(a) Convert it into matrix representation [Allocation, Request and Available].
(b) Draw the equivalent wait for graph and list the cycles.
(c) Is there a deadlock in the system? If Yes, which are the process involved in deadlock? If No, give the set of all possible safe sequences.



Solution:

(a) The matrix representation [Allocation, Request, and Available] is as follows.

Allocation Matrix: [1 mark]

	R1	R2	R3	R4	R5	R6
P1	1	0	0	0	1	0
P2	1	0	0	1	0	0
P3	0	0	0	0	0	1
P4	0	1	0	1	0	0
P5	0	0	1	0	0	1

Request Matrix: [1 mark]

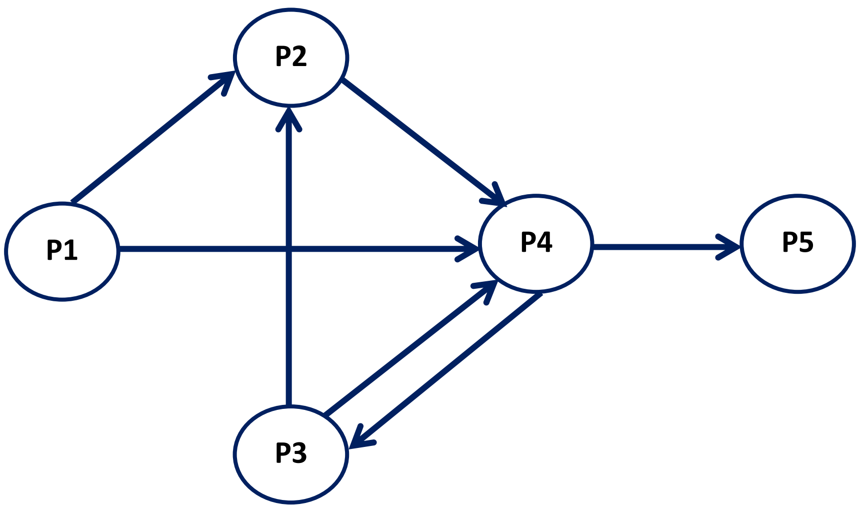
	R1	R2	R3	R4	R5	R6
P1	0	0	0	1	0	0
P2	0	1	0	0	0	0
P3	0	0	0	1	0	0
P4	0	0	0	0	0	1
P5	0	0	0	0	0	0

Available Matrix: [0.5 mark]

	R1	R2	R3	R4	R5	R6
	0	0	1	0	0	0

(b) Wait-for Graph: **[1.5 marks for neat labelled graph.]**

For every {$P_i \rightarrow R_k$ and $R_k \rightarrow P_j$ }in RAG, we have a $P_i \rightarrow P_j$ in WFG



(c) **No**, there will be no deadlock in the system. Initially P5 will complete and then P4. Rest three can then be completed in any order. There are 6 possible safe states as follows: **[0.5 marks (for mentioning no deadlock) +1.5 (listing 6 sequences) = 2 marks]**

P5 – P4 – P3 – P2 – P1
P5 – P4 – P3 – P1 – P2
P5 – P4 – P2 – P3 – P1
P5 – P4 – P2 – P1 – P3
P5 – P4 – P1 – P3 – P2
P5 – P4 – P1 – P2 – P3

Question 6:

In a system there are two types of processes: type A processes and type B processes. All processes of type A execute the same code, and all processes of type B execute the same code. The code for each process type is shown below. Here, X and Y are general semaphores. X is initialized to 2, and Y is initialized to 0. Suppose three processes of type A and two processes of type B are brought into execution simultaneously. Once a process starts, assume that there is no context switching before its completion. Answer the following two questions:

Process A	Process B

Wait (X);
Signal (Y);
	Wait (Y);
	Wait (Y);
	Signal (X);
	Signal (Y);

(a) Is it possible for processes to finish in the order of AABAB? If so, show an execution sequence showing the values of semaphores X and Y in every step. If not, explain why?

(b) Is it possible for processes to finish in the order AABBA? If so, show an execution sequence showing the values of semaphores X and Y in every step. If not, explain why?

Solution:

(a) The following is an execution sequence showing that AABAB is possible:
[1 mark (for mentioning possible) +2 (illustration) = 3 marks]

A1	**A2**	**B1**	**A3**	**B2**	**Semaphore X**	**Semaphore Y**
					2	0
Wait (X)					1	0
Signal (Y)					1	1
	Wait (X)				0	1
	Signal (Y)				0	2
		Wait (Y)			0	1
		Wait (Y)			0	0
		Signal (X)			1	0
		Signal (Y)			1	1
			Wait (X)		0	1
			Signal (Y)		0	2
				Wait (Y)	0	1
				Wait (Y)	0	0
				Signal (X)	1	0
				Signal (Y)	1	1

This execution sequence shows clearly that processes A1, A2 and A3 in group A, and processes B1 and B2 in group B can indeed produce the order AABAB.

(b) The following is an execution sequence showing that AABBA is not possible:
[1 mark (for mentioning not possible) +2 (illustration) = 3 marks]

A1	**A2**	**B1**	**B2**	**A3**	**Semaphore X**	**Semaphore Y**
					2	0
Wait (X)					1	0
Signal (Y)					1	1
	Wait (X)				0	1
	Signal (Y)				0	2
		Wait (Y)			0	1
		Wait (Y)			0	0
		Signal (X)			1	0
		Signal (Y)			1	1
			Wait (Y)		1	0
			Wait (Y) $\longrightarrow$			**BLOCKED**
			Signal (X)			
			Signal (Y)			

The sequence AABBA is not possible. From the above execution sequence, after execution of two A (A1 followed by A2) semaphores X and Y have counters 0 and 2, respectively. If B1 follows then, X and Y will be 1 and 1, respectively. Because Y value is 1, the next B (ie B2) can only pass its first Wait (Y) and will be blocked by its second Wait (Y), and, as a result, B2 and A3 cannot complete its execution. Hence, the order of AABBA is impossible.