Dec 4, 2024

CS 350 DISCUSSION 12

Bankers Algorithm and Tips for Challenge

Problem 6

Consider as system with 5 processes $P_1, \ldots P_5$ sharing 3 resources $R_1, \ldots, R_3$.

(a) **[5 points]** The immutable system parameters are provided in Table 1, while the state of the system at a given point in time is provided in Table 2. Complete the tables with the missing parameters.

		Parameter	Resources		
			R_1	R_2	R_3
		$R(k)$			
Processes	P_1	$C_1(k)$	2	1	3
	P_2	$C_2(k)$	5	4	2
	P_3	$C_3(k)$	5	5	3
	P_4	$C_4(k)$	6	6	7
	P_5	$C_5(k)$	4	4	3

Table 1: Static parameters for the considered system.

		Parameter	Resources			Parameter	Resources		
			R_1	R_2	R_3		R_1	R_2	R_3
		$V(k)$	2	1	1				
Processes	P_1	$A_1(k)$	0	1	2	$N_1(k)$			
	P_2	$A_2(k)$	0	0	0	$N_2(k)$			
	P_3	$A_3(k)$	3	2	0	$N_3(k)$			
	P_4	$A_4(k)$	5	2	2	$N_4(k)$			
	P_5	$A_5(k)$	0	0	3	$N_5(k)$			

Table 2: System state for considered system.

- $R(k)$: total amount of resource R_k present in the system. The idea is that each resource exists in the system in a certain amount. For instance, $R(k = \text{main memory})$ =4 GB; $R(k = \text{disk space})$ = 1 TB; and so on. Mutually exclusive resources have availability $R(k) = 1$.
- $C_i(k)$: total amount of resource R_k that process P_i will ever need during its execution. It can be though as the amount of R_k that P_i claims it needs to execute. This quantity is simply declared by P_i before starting its execution.
- $A_i(k)$: amount of R_k currently allocated (i.e. granted) to process P_i. Upon P_i's start, $A_i(k) = 0$ for all k. Obviously, it must hold that $A_i(k) \leq C_i(k) \leq R(k)$.
- $V(k)$: total amount of resource R_k available at the current time in the system. This quantity can be calculated as: $V(k) = R(k) - (A_1(k) + A_2(k) + \cdots + A_N(k))$.
- $N_i(k)$: amount of R_k that process P_i currently needs to complete its execution. $N_i(k)$ can be computed as follows: $N_i(k) = C_i(k) - A_i(k)$.

Example Final Question

(a) [5 points] The immutable system parameters are provided in Table 1, while the state of the system at a given point in time is provided in Table 2. Complete the tables with the missing parameters.

		Parameter	Resources R_1	R_2	R_3
		$R(k)$	10	6	8
Processes	P_1	$C_1(k)$	2	1	3
	P_2	$C_2(k)$	5	4	2
	P_3	$C_3(k)$	5	5	3
	P_4	$C_4(k)$	6	6	7
	P_5	$C_5(k)$	4	4	3

Table 3: Static parameters for the considered system.

		Parameter	Resources R_1	R_2	R_3	Parameter	Resources R_1	R_2	R_3
		$V(k)$	2	1	1				
Processes	P_1	$A_1(k)$	0	1	2	$N_1(k)$	2	0	1
	P_2	$A_2(k)$	0	0	0	$N_2(k)$	5	4	2
	P_3	$A_3(k)$	3	2	0	$N_3(k)$	2	3	3
	P_4	$A_4(k)$	5	2	2	$N_4(k)$	1	4	5
	P_5	$A_5(k)$	0	0	3	$N_5(k)$	4	4	0

Table 4: System state for considered system.

(b) [6 points] Determine if the current state would be deemed safe by the Banker's Algorithm for deadlock avoidance. Provide your reasoning for full marks.

		Parameter	Resources		
			R_1	R_2	R_3
		$R(k)$	10	6	8
Processes	P_1	$C_1(k)$	2	1	3
	P_2	$C_2(k)$	5	4	2
	P_3	$C_3(k)$	5	5	3
	P_4	$C_4(k)$	6	6	7
	P_5	$C_5(k)$	4	4	3

Table 3: Static parameters for the considered system.

		Parameter	Resources			Parameter	Resources		
			R_1	R_2	R_3		R_1	R_2	R_3
		$V(k)$	2	1	1				
Processes	P_1	$A_1(k)$	0	1	2	$N_1(k)$	2	0	1
	P_2	$A_2(k)$	0	0	0	$N_2(k)$	5	4	2
	P_3	$A_3(k)$	3	2	0	$N_3(k)$	2	3	3
	P_4	$A_4(k)$	5	2	2	$N_4(k)$	1	4	5
	P_5	$A_5(k)$	0	0	3	$N_5(k)$	4	4	0

Table 4: System state for considered system.

(b) [6 points] Determine if the current state would be deemed safe by the Banker's Algorithm for deadlock avoidance. Provide your reasoning for full marks.

P1 ✓ → V(k) = 2 1 1 = 2 2 3
+ 0 1 2

P2 ×
P3 × P4 ×
P5 ×

Only P1 is safe.

P2, P3, P4, and P5 are all unsafe since they all cannot complete the execution.

Therefore, not safe.

		Parameter	Resources		
			R_1	R_2	R_3
		$R(k)$	10	6	8
Processes	P_1	$C_1(k)$	2	1	3
	P_2	$C_2(k)$	5	4	2
	P_3	$C_3(k)$	5	5	3
	P_4	$C_4(k)$	6	6	7
	P_5	$C_5(k)$	4	4	3

Table 3: Static parameters for the considered system.

		Parameter	Resources			Parameter	Resources		
			R_1	R_2	R_3		R_1	R_2	R_3
		$V(k)$	2	1	1				
Processes	P_1	$A_1(k)$	0	1	2	$N_1(k)$	2	0	1
	P_2	$A_2(k)$	0	0	0	$N_2(k)$	5	4	2
	P_3	$A_3(k)$	3	2	0	$N_3(k)$	2	3	3
	P_4	$A_4(k)$	5	2	2	$N_4(k)$	1	4	5
	P_5	$A_5(k)$	0	0	3	$N_5(k)$	4	4	0

Table 4: System state for considered system.

c) [7 points] While the system is in the state described by Table 1 and 2, P3 submits an allocation request for 0 units of R1, 1 unit of R2, and 0 unit of R3. Can the request be safely granted? You may use Table 5 to answer the question.

		Parameter	Resources			Parameter	Resources		
			R_1	R_2	R_3		R_1	R_2	R_3
		$V(k)$							
Processes	P_1	$A_1(k)$				$N_1(k)$			
	P_2	$A_2(k)$				$N_2(k)$			
	P_3	$A_3(k)$				$N_3(k)$			
	P_4	$A_4(k)$				$N_4(k)$			
	P_5	$A_5(k)$				$N_5(k)$			

Table 5: System State.

c) [7 points] While the system is in the state described by Table 1 and 2, P3 submits an allocation request for 0 units of R1, 1 unit of R2, and 0 unit of R3. Can the request be safely granted? You may use Table 5 to answer the question.

The new state if we try to pretend to grant the request is reported in *Table 6*. The initial **V(k) = [2, 0, 1]**. With this, P1 can complete.

		Parameter	Resources			Parameter	Resources		
			R_1	R_2	R_3		R_1	R_2	R_3
		$V(k)$	2	0	1				
Processes	P_1	$A_1(k)$	0	1	2	$N_1(k)$	2	0	1
	P_2	$A_2(k)$	0	0	0	$N_2(k)$	5	4	2
	P_3	$A_3(k)$	3	3	0	$N_3(k)$	2	1	3
	P_4	$A_4(k)$	5	2	2	$N_4(k)$	1	4	5
	P_5	$A_5(k)$	0	0	3	$N_5(k)$	4	4	0

Table 6: *Pretend* system state for considered system.

No, since it could not be granted with the previous state, so adding more allocated resources will not help since P2, P3, P4 and P5 were not safe.

Tips for Extra Credit Challenge

You have a lot of options!!!

1) 'Hack' CodeBuddy into giving you what you want. We don't elaborate too much on this :) It's up to you to figure it out.

2) If you want to do it the fair way, focus on the following:

a) Try different schedulers (other than FIFO and SJN)

b) Optimize your memory, runtime and try and be as efficient as possible.

c) Any .c and .h file can be modified. Feel free to modify or even improve any of the image processing code.