Operating Systems - Review Questions 2

Deadline: Nov. 25, 2023
Group A&H

Students:
Hiba Qutbuddin Habib
Ayah Babiker

1. **What is the meaning of the term busy waiting? Can busy waiting be avoided altogether? Explain your answer.**

   Busy waiting is a technique where a process repeatedly checks a condition before moving forward. Avoiding it entirely can be challenging. While you can inhibit interrupts in a uniprocessor system during certain operations, it's not feasible in multiprocessor systems, making complete avoidance difficult. This inefficient approach should be minimized when possible.

2. **Show that, if the wait() and signal() semaphore operations are not executed atomically, then mutual exclusion may be violated.**

   If the wait() and signal() semaphore operations are not executed atomically, mutual exclusion can be violated. This is because the semaphore values are modified by these operations: wait() decreases the value, and signal() increases it. When these operations are not atomic, multiple processes can potentially modify the semaphore simultaneously, which can lead to a situation where more than one process accesses a critical section concurrently, violating mutual exclusion.

3. **Consider the following snapshot of a system, and answer the following questions using the banker's algorithm:**

	Allocation	Max	Available
	A B C D	A B C D	A B C D
T_0	0 0 1 2	0 0 1 2	1 5 2 0
T_1	1 0 0 0	1 7 5 0	
T_2	1 3 5 4	2 3 5 6	
T_3	0 6 3 2	0 6 5 2	
T_4	0 0 1 4	0 6 5 6	

a. What is the content of the matrix Need?

Max - Allocation = Need

	Need (A)	Need (B)	Need (C)	Need (D)
T0	0 - 0 = 0	0 - 0 = 0	1 - 1 = 0	2 - 2 = 0
T1	1 - 1 = 0	7 - 0 = 7	5 - 0 = 5	0 - 0 = 0
T2	2 - 1 = 1	3 - 3 = 0	5 - 5 = 0	6 - 4 = 0
T3	0 - 0 = 6	6 - 6 = 0	5 - 3 = 0	2 - 2 = 0
T4	0 - 0 = 0	6 - 0 = 6	5 - 1 = 4	6 - 4 = 2

b. Is the system in a safe state?

- Work = Available = 1 5 2 0
- Finish = false
- Need0 < work
- Work = 1 5 2 0 + 0 0 1 2 = 1 5 3 2
- finish[0]= true
- Need1 > work
- finish[1]= still false
- Need2 < work
- finish[2]= true
- Work = 1 5 3 2 + 1 3 5 4 = 2 8 8 6
- Need3 < work
- Work = 2 8 8 6 + 0 6 3 2 = 2 14 11 8
- Finish[3] = true
- Need4 < work
- Work = 2 14 11 8 + 0 0 1 4 = 2 14 13 12
- Finish[4]= true
- Need1 < work
- Work = 2 14 13 12 + 1 0 0 0 = 3 14 13 12
- finish[1]= true
- Finish=true

Since there is a a safe sequence the system is safe and the sequence is T0, T2, T3, T4, T1

c. If a request from thread T_1 arrives for (0, 4, 2, 0), can the request be granted immediately?

Yes it can be granted, the request is less than the need for T0 and the request is also less than available.

4. Consider the following resource-allocation policy. Requests for and releases of resources are allowed at any time. If a request for resources cannot be satisfied because the resources are not available, then we check any threads that are blocked waiting for resources. If a blocked thread has the desired resources, then these resources are taken away from it and are given to the requesting thread. The vector of resources for which the blocked thread is waiting is increased to include the resources that were taken away.

For example, a system has three resource types, and the vector Available is initialized to (4, 2, 2). If thread T_0 asks for (2, 2, 1), it gets them. If T_1 asks for (1, 0, 1), it gets them. Then, if T_0 asks for (0, 0, 1), it is blocked (resource not available). If T_2 now asks for (2, 0, 0), it gets the available one (1, 0, 0), as well as one that was allocated to T_0 (since T_0 is blocked). T_0's Allocation vector goes down to (1, 2, 1), and its Need vector goes up to (1, 0, 1).

a. Can deadlock occur? If you answer "yes", give an example. If you answer "no", specify which necessary condition cannot occur.

A deadlock cannot occur since this resource policy allows preemption which means that for example if resources are preempted from a blocked thread they can be allocated to other requests. Preemption means that deadlock cannot happen; the threads will not be waiting for each other and stopping the other for accessing the resources they need.

b. Can indefinite blocking occur? Explain your answer.

Yes, indefinite blocking can occur when a process continuously resources it needs, and those resources are never allocated to it, leading to an indefinite waiting period.