CS471: Operating System Concepts
Module 4: Homework #4
Solution

Points: 20

Question 1 [Points 8] Consider the following snapshot of a system:

	Allocation	*Max*	*Available*	*Need*
	A B C D	*A B C D*	*A B C D*	*A B C D*
*P*0	2 0 0 1	4 2 1 2	3 3 2 1	2 2 1 1
*P*1	3 1 2 1	5 2 5 2		2 1 3 1
*P*2	2 1 0 3	2 3 1 6		0 2 1 3
*P*3	1 3 1 2	1 4 2 4		0 1 1 2
*P*4	1 4 3 2	3 6 6 5		2 2 3 3

Answer the following questions using the banker's algorithm:

a. Illustrate that the system is in a safe state by demonstrating an order in which the processes may complete.

Ans: First, we compute the Need as shown above by computing Need = Max – Allocation.

Now, let us see which of the above needs may be met by the Available resources.

- P0's Need are satisfied. So assume we have allocated these resources to P0. P0 completes execution and then releases its resources. Now Available= <3 3 2 1> + <2 0 0 1> = <5 3 2 2>.
- Now, P3's Need are satisfied. So assume we have allocated these resources to P3. P3 completes execution and then releases its resources. Now Available= <5 3 2 2> + <1 3 1 2> = <6 6 3 4>.
- Now, P3's Need are satisfied. So assume we have allocated these resources to P3. P3 completes execution and then releases its resources. Now Available= <5 3 2 2> + <1 3 1 2> = <6 6 3 4>.
- Now, P1, P2, or P4's Need may be satisfied. Assume we have allocated resources to P1. P1 completes execution and then releases its resources. Now Available= <6 6 3 4> + <3 1 2 1>= <9 7 5 5>.
- Now, P2 or P4's Need may be satisfied. Assume we have allocated resources to P2. P2 completes execution and then releases its resources. Now Available= <9 7 5 5>+ <2 1 0 3>= <11 8 5 8>.
- Now P4's Needs may be satisfied. It completes execution and releases its resources, making Available = <11 8 5 8> + <1 4 3 2> = >12 12 8 10>

Since there is at least one sequence <P0, P3, P1, P2,P4> that can successfully complete execution, there is no deadlock.

In fact, With <P0, P3> as prefix, any other sequence with arbitrary ordering of P1, P2, P4 will be possible through the Banker's algorithm.

b. If a request from process *P*1 arrives for (1, 1, 0, 0), can the request be granted immediately?

Ans: To answer this question, first let us assume that the request (1, 1, 0, 0) is granted. Then Available=<3 3 2 1> - <1 1 0 0> = < 2 2 2 1>.

- P0 can still complete with the Available. After it completes, Available = < 2 2 2 1> + <2 0 0 1> = <4 2 2 2>.
- P3 can still finish with the Available. After it completes, Available = <4 2 2 2> + <1 3 1 2> = <5 5 3 4>.
- P1, P2, or P4 can finish in any order as shown above.

Since there is a possible sequence in which all processes can finish even after granting P1's (1 1 0 0) request, the request **may be granted immediately**.

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

Ans: To answer this question, first let us assume that the request (0, 0, 2, 0) is granted. Then Available=<3 3 2 1> - <0 0 2 0> = < 3 3 0 1>.

None of the processes P0-P4 can finish with the new Available. So we **should defer** granting resources to this request.

Question 2. [Points 6] Consider a system consisting of four resources of the same type (i.e., four instances of one type of resource) that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock free.

Ans: So Available = <4>. Max request for each process = <2>.

Several cases exist: Case 1: Each process requests for one resource. They can be granted. We still have 1 Available resource. So when one of the three processes requests, that request can be granted and that process can finish and release its resources, making Available=2. This can continue with other processes. So all can finish and there is no deadlock.

Case 2. Two processes acquire 2 resources each. So when they finish, they release resources and Available=4 which will enable other two processes to finish. There is no deadlock.

Case 3. Only one process acquires 2 resources. Situation is the same as case 2.

Thus, the system is deadlock free.

Question 3 [Points 6] Consider the version of the dining-philosophers problem in which the chopsticks are placed at the center of the table and any two of them can be used by a philosopher. Assume that requests for chopsticks are made one at a time. Describe a simple rule for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

Ans: Let us assume that there are N dining philosophers. There are a total of N chopsticks in the center at the beginning. So Available=N. Max=2 since each philosopher needs 2 chopsticks to finish.

Each request can ask for only one chopstick.

We can use Banker's rule for this as follows.

Grant a request for a chopstick from a Philosopher Pi only when:

1. The number of chopsticks in the center is ≥1, AND
2. There is at least one philosopher who can finish (have 2 chopsticks) after Pi's request is granted.

Illustration: Let there be 4 Philosophers. So there are 4 chopsticks in the center. So Available=4. Right now, no one is holding a chopstick.

P1 makes a request for a chopstick. Let us check the rule. Available ≥1 and after the request is granted Available=3. With this, P1, P2, P3, or P4 can finish.

So the rule is satisfied and hence request may be granted.