Tecnológico de Costa Rica 12 de setiembre de 2017

Escuela de Ingeniería en Computación Samantha Arburola León
Principios de Sistemas Operativos 2013101697

Operating System Concepts

Chapter 7 - Practice Exercises

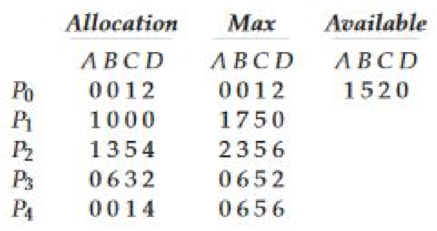
7.1. List three examples of deadlocks that are not related to a computer-system environment

- Two persons climb a single-lane stairs from opposite directions.
- A car going out from parking while another car is going in.
- Two cars traveling toward each other on the same track; in a rural route.

7.2 Suppose that a system is in an unsafe state. Show that it is possible for the processes to complete their execution without entering a deadlocked state

An unsafe state may not necessarily lead to deadlock, it just means that we can not guarantee that deadlock will not occur

7.3 Consider the following snapshot of a system



	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker 's algorithm:

a. What is the content of the matrix Need ?
b. Is the system in a safe state?
c. If a request from process P1 arrives for (0,4,2,0), can the request be granted immediately?

a. Need for processes P0 through P4 respectively are
(0, 0, 0, 0),
(0, 7, 5, 0),
(1, 0, 0, 2),
(0, 0, 2, 0),
(0, 6, 4, 2).

b. Yes. With Available equal to (1, 5, 2, 0), process P0 or P3 could run.
c. This results in the value of Available being (1, 1, 0, 0). One ordering of processes that can finish is P0, P2, P3, P1, and P4.

7.4 A possible method for preventing deadlocks is to have a single, higher- order resource that must be requested before any other resource. For example, if multiple threads attempt to access the synchronization objects A ··· E, deadlock is possible. (Such synchronization objects may include mutexes, semaphores, condition variables, and the like.) We can prevent the deadlock by adding a sixth object F. Whenever a thread wants to acquire the synchronization lock for any object A ··· E, it must first acquire the lock for object F. This solution is known as containment: the locks for objects A ··· E are contained within the lock for object F. Compare this scheme with the circular-wait scheme of Section 7.4.4

It is better to define a locking policy with a narrow a scope as possible.

7.5 Prove that the safety algorithm presented in Section 7.5.3 requires an order of m × n^2 operations.

```
for(int 1 =0; i < n; i++){

    for (int j = 0; j < n; j++){
      if (!finish[j]){
           boolean temp = true;
           for(int k = 0; k < m; k++){
                if (need[j][k] > work[k])
                     temp = false;
           }

           if (temp){
                finish[j] = true;
                for(int x = 0; x < m; x++)
                     work[x] += work[j][x];
           }
      }
    }
}
```

7.6 Consider a computer system that runs 5,000 jobs per month and has no deadlock-prevention or deadlock-avoidance scheme. Deadlocks occur about twice per month, and the operator must terminate and rerun about ten jobs per deadlock. Each job is worth about two dollars (in CPU time), and the jobs terminated tend to be about half done when they are aborted. A systems programmer has estimated that a deadlock-avoidance algorithm (like the banker's algorithm) could be installed in t are starving? If you answer "yes", explain how it can. If you answer "no", explain hohe system with an increase of about 10 percent in the average execution time per job. Since the machine currently has 30 percent idle time, all 5,000 jobs per month could still be run, although turnaround time would increase by about 20 percent on average.

a. What are the arguments for installing the deadlock-avoidance algorithm?
b. What are the arguments against installing the deadlock-avoidance algorithm?

Un argumento para instalar deadlock del interbloqueo en el sistema es que podríamos asegurar que el interbloqueo nunca ocurriera. Además, a pesar del aumento del tiempo de respuesta, todos los 5.000 puestos de trabajo podrían seguir ejecutándose. Un argumento contra la instalación de software de prevención de bloqueo es que los bloqueos ocurren con poca frecuencia y cuestan poco cuando ocurren.

7.7 Can a system detect that some of its processes w the system can deal with the starvation problem

One way of detecting starvation could identify a period of time -T- that is considered unreasonable. If the time elapsed exceeds T, and considers that the process is starved.

7.8 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 processes that are blocked waiting for resources. If a blocked process has the desired resources, then these resources are taken away from it and are given to the requesting process. The vector of resources for which
the blocked process 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 process P0 asks for (2,2,1), it gets them. If P1 asks for (1,0,1), it gets them. Then, if P0 asks for (0,0,1), it is blocked (resource not available). If P2 now asks for (2,0,0), it gets the available one (1,0,0), as well as one that was allocated to P0 (since P0 is blocked). P0'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.
b. Can indefinite blocking occur? Explain your answer.

a. Deadlock cannot occur because preemption exists.
b. Yes. A process may never acquire all the resources it needs if they are continuously preempted.

7.9 Suppose that you have coded the deadlock-avoidance safety algorithm and now have been asked to implement the deadlock-detection algorithm. Can you do so by simply using the safety algorithm code and
redefining Max_i = $Waiting_i$ + $Allocation_i$, where Waiting i is a vector specifying the resources for which process i is waiting and Allocation i is as defined in Section 7.5? Explain your answer.
Yes. Max vector represents maximum request a process may make, we use Need matrix for calculating the safety algorithm
Max = Need + Alllocation

7.10 Is it possible to have a deadlock involving only one single-threaded process? Explain your answer
Deadlock with one process is not possible
A deadlock situation can arise if the following four conditions hold simultaneously in a system.It is not possible to have circular wait with only one process, thus failing a necessary condition for Circular wait. There is no second process to form a circle with the first one, by the way it is not possible to have a deadlock involving only one process