CPSC 457 - Assignment 4
Winter 2018

Lamess Kharfan
ID: 10150607
Tutorial: T07

Question 1- Provide a simple example of a system in an unsafe state, where the processes could finish without entering a deadlock. Show the state, and two sequences of execution: one leading to a deadlock and the other leading to completion of all processes.

Example of an unsafe state:
There are 3 processes.
One resource type exists and there are 12 instances of this resources.

The state of the system is as follows:

Process	Allocation	Max	Available	Need
P1	5	11	3	6
P2	2	4		2
P3	2	9		7

Execution sequence leading to deadlock:

1. Scheduler runs process P2 exclusively, which eventually requests it maximum resources, being 4, by requesting an additional 2 resources, which are available. When P2 completes, it will release all of its resources, making 5 available total. P2 now has 0 resources allocated to it.
2. Now each of the active processes, P1 and P3 cannot run because P1 requires 6 more resources and P3 requires 7 more resources in case they need to request their maximum resources immediately, and neither of them can be scheduled.

➢ The system is now in deadlock, and the state is unsafe because there is no sequence of execution that guarantees completion.

Execution sequence leading to completion of all processes:

1. Process P3 releases one of it's resources. It now has 1 allocated to it, and "Need" changes to 8 for P3. Now, 4 resources in total are available.
2. Process P2 runs, and eventually requests an additional 2 resources, which are available. P2 now has 4 resources allocated to it. Process P2 eventually completes and releases its resources. Now P2 has 0 resources allocated to it and there is a total of 6 available.
3. Process P1 runs, and eventually requests an additional 6 resources, which are available. P1 now has 11 resources allocated to it. Process P1 eventually completes and releases its resources. Now P1 has 0 resources allocated to it and there is a total of 11 resources available.
4. Process P2 runs, and eventually requests an additional 8 resources, which are available. P3 now has 8 resources allocated to it. Process P3 eventually completes and releases its resources. Now P3 has 0 resources allocated to it and there is a total of 12 resources available.

➢ All processes were able to finish in this execution sequence of the state of the system, even in the case that they request their maximum number of resources immediately.

Question 2 – What is the smallest value of ‘x’ that can keep the system safe? Once you find the ‘x’ value, find a safe execution order using the safety algorithm. Include a step-by-step walkthrough of the safety algorithm.

The smallest value of ‘x’ that can keep the system in a safe state is 1. This is because P1, P2, and P3 have less than the maximum amount of this resource allocated to them, and if there is at least 1 of this resource available, there exists a scheduling order in which every process is guaranteed to run to completion, keeping it in a safe state.

Process	Allocation	Maximum	Available	Need
P0	1 0 2 1 1	1 1 2 1 3	0 0 1 1 2	0 1 0 0 2
P1	2 0 1 1 0	2 2 2 1 0		0 2 1 0 0
P2	1 1 0 1 0	2 1 3 1 0		1 0 3 0 0
P3	1 1 1 1 0	1 1 2 2 1		0 0 1 1 1

Step-by-step walk-though of safety algorithm:

//Initialize Temporary Vectors
Work = Available
Finish[0] = false
Finish[1] = false
Finish[2] = false
Finish[3] = false

//Find i such that finish[i] = false and Need[i] <= Work
//Update work and finish, go to step 2.
i = 3
finish[3] = false and need[3] <= work (true)
Work = Work + Allocation[3]
= (0 0 1 1 2) + (1 1 1 1 0) = (1 1 2 2 2)
Finish[3] = true

i = 0
finish[0] = false and need[0] < = work (true)
work = work + allocation[0]
= (1 1 2 2 2) + (1 0 2 1 1) = (2 1 4 3 3)
Finish[0] = true

i = 1
finish[1] = false and need[1] < = work (true)
work = work + allocation[1]
= (2 1 4 3 3) + (2 0 1 1 0) = (4 1 5 4 3)
Finish[1] = true

i = 2
finish[2] = false and need[2] < = work (true)
work = work + allocation[2]
= (4 1 5 4 3) + (1 1 0 1 0) = (5 2 5 5 3)
Finish[1] = true

//If finish[i] == true for all i, return true
Finish[i] == true for all I is true
return true

In summary:
1. P3: available = (0 0 1 1 2) + (1 1 1 1 0) → (1 1 2 2 2)
2. P0: available = (1 1 2 2 2) + (1 0 2 1 1) → (2 1 4 3 3)
3. P1: available = (2 1 4 3 3) + (2 0 1 1 0) → (4 1 5 4 3)
4. P2: available = (4 1 5 4 3) + (1 1 0 1 0) → (5 2 5 5 3)

Execution sequence: P3, P0, P1, P2.