Operating Systems: Homework #4

Due on March 23, 2016 at 11:59pm

Professor Qu
Monday & Wednesday 3:30pm — 5:17pm

Nicholas Land

Problem 1

Answer the following questions

(a) Describe a real-life deadlock situation. Explain why it satisfies the four necessary conditions (mutual exclusion, hold-and-wait, non-preemption, circular wait). How do people recover from that situation? Upon recovery, which condition becomes false?

(b) Give an example, where the system is not in a safe state, but if the processes of the system are allowed to be executed, then they will be successfully completed.

SOLUTION

(a) An example of a real life deadlock would be something like a person trying to get a professional job, but are unable to because they lack the experience to get the job. However, to get the experience, they need to have a job. This satisfies the four condtions of deadlock as follows:

- MUTUAL EXCLUSION
  One of the things (get a professional job) is unable to occur because it is dependent on the other (experience), and visa-versa.
- HOLD AND WAIT
  This is essentially the same as the previous bullet point.
- NO PREEMPTION
  This holds because you can't get one without the other. As a result, none of the two 'proceeses‘ could preempt one another.
- CIRCULAR WAIT
  This condition is satisfied because if you want the job, you need the experience, if you want the expirenece, you need the job. That is a continuous cycle.

People are able to recover from such deadlock by getting an internship, or an apprenticeship which will give them the experience that they need to get the job. The condition that becomes false is the circular wait. Now, the person has experience and is able to obtain the professional job.

(b) Like in the previous example I provided, if the user were able to obtain an internship, then the processes could execute as normal, and all four condtions of deadlock would be satisfied. In this case deadlock would not occur.

Problem 2

Consider the following snapshot of a system (P=Process, R=Resource) :

Available			
R_a	R_b	R_c	R_d
1	5	2	0

Problem 2 continued on next page...

Maximum Demand				
	R_a	R_b	R_c	R_d
P_0	0	3	1	2
P_1	1	7	5	0
P_2	2	3	5	6
P_3	0	6	5	2
P_4	0	6	5	6

Current Allocation				
	R_a	R_b	R_c	R_d
P_0	0	0	1	2
P_1	1	0	0	0
P_2	1	3	5	4
P_3	0	6	3	2
P_4	0	0	1	4

Answer the following questions using banker's algorithm:

a) Calcualtes the *Needs* matrix:

Needs				
	R_a	R_b	R_c	R_d
P_0				
P_1				
P_2				
P_3				
P_4				

b) Is the system in a safe state? If so, show how you derive a safe order with Safety Algorithm in which the processes can run. Show the different values of the work vector after each iteration. What is the sequence of processes that the algorithm implicitly created?

SOLUTION

a) Needs Matrix:

Needs				
	R_a	R_b	R_c	R_d
P_0	0	3	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

b) Derived safe state:
Initial Work:

Work			
R_a	R_b	R_c	R_d
1	5	2	0

Derived safe order:
$<P_0>$

Work			
R_a	R_b	R_c	R_d
1	5	3	2

Problem 2 continued on next page...

Derived safe order:
<P_0, P_2>

Work			
R_a	R_b	R_c	R_d
2	8	8	6

Derived safe order:
<P_0, P_2, P_3>

Work			
R_a	R_b	R_c	R_d
2	14	11	8

Derived safe order:
<P_0, P_2, P_3, P_4>

Work			
R_a	R_b	R_c	R_d
2	14	12	12

Derived safe order:
<P_0, P_2, P_3, P_4, P_1>

Work			
R_a	R_b	R_c	R_d
3	14	12	12