Deadlocks

Week 13 – Tutorial

Problem 6.8

- Is it possible that a resource deadlock involves multiple units of one type and a single unit of another? If so, give an example

Problem 6.8 – Solution

- Yes, it is possible. For example, two processes are both allocated memory cells in a real memory system (assume that swapping is not supported):
  - The first process locks another resource - perhaps a data cell
  - The second process requests the locked data and is blocked. The first process needs more memory in order to execute the code to release the data. Assuming that no other processes in the system can complete and release memory cells, a deadlock exists in the system

Work in pairs
Think-pair-share

Problem 6.14 (1/2)

- Consider the following state of a system with four processes, P1, P2, P3, and P4, and five types of resources, RS1, RS2, RS3, RS4, and RS5:

C =

0	1	1	1	2
0	1	0	1	0
0	0	0	0	1
2	1	0	0	0

R =

1	1	0	2	1
0	1	0	2	1
0	2	0	3	1
0	2	1	1	0

E = (24144)

A = (01021)

Problem 6.14 (2/2)

- Using the deadlock detection algorithm described in Section 6.4.2, show that there is a deadlock in the system. Identify the processes that are deadlocked

Problem 6.14 – Solution

- First, the set of unmarked processes, P = (P1 P2 P3 P4)
  - R1 is not less than or equal to A
  - R2 is less than A; Mark P2; A = (0 2 0 3 1); P = (P1 P3 P4)
  - R1 is not less than or equal to A
  - R3 is equal to A; Mark P3; A = (0 2 0 3 2); P = (P1 P4)
  - R1 is not less than or equal to A
  - R4 is not less than or equal to A
- Processes P1 and P4 remain unmarked that means they are deadlocked

Problem 6.15

- Explain how the system can recover from the deadlock in previous problem using:
  A. Recovery through preemption
  B. Recovery through rollback
  C. Recovery through killing processes

Problem 6.15 – Solution (1/3)

- Recovery through preemption:
  - After processes P2 and P3 complete, process P1 can be forced to preempt 1 unit of RS3.
  - This will make A=(0 2 1 3 2), and allow process P4 to complete.
  - Once P4 completes and releases its resources, P1 may complete

Problem 6.15 – Solution (2/3)

- Recovery through rollback:
  - Rollback P1 to the state checkpointed before it acquired RS3

Problem 6.15 – Solution (3/3)

- Recovery through killing processes:
  - Kill P1

Work in pairs
Think-pair-share

Problem 6.20

- Can a system be in a state that is neither deadlocked nor safe? If so, give an example. If not, prove that all states are either deadlocked or safe

Problem 6.20 – Solution (1/2)

- There are states that are neither safe nor deadlocked, but which lead to deadlocked states. As an example, suppose we have four resources:
  - 3 tapes,
  - 2 plotters,
  - 4 scanners, and
  - 2 CD-ROMs,
- and three processes (A,B,C) competing for them

Problem 6.20 – Solution (2/2)

- We could have the following situation:

	Has	Needs	Available
A:	2 0 0 0	1 0 2 0	0 1 2 1
B:	1 0 0 0	0 1 3 1	
C:	0 1 2 1	1 0 1 0	

- This state is not deadlocked because many actions can still occur, for example, A can still get two plotters. However, if each process asks for its remaining requirements, we have a deadlock

Problem 6.22

- A system has **two** processes and **three** identical resources. Each process needs a maximum of two resources. Is deadlock possible? Explain your answer

Problem 6.22 – Solution

- The system is deadlock free. Suppose that each process has one resource. There is one resource free. Either process can ask for it and get it, in which case it can finish and release both resources
- Consequently, deadlock is impossible

Problem 6.23

- Consider the previous problem again, but now with **p** processes each needing a maximum of **m** resources and a total of **r** resources available.
- What condition must hold to make the system deadlock free?

Problem 6.23 – Solution

- If a process has *m* resources it can finish and cannot be involved in a deadlock
- The worst case is where every process has $m - 1$ resources and needs another one. If there is one resource left over, one process can finish and release all its resources, letting the rest finish too
- The condition for avoiding deadlock is

$$r \geq p(m - 1) + 1$$

Problem 6.25

- The banker's algorithm is being run in a system with ***m*** resource classes and ***n*** processes
- In the limit of large *m* and *n*, the number of operations that must be performed to check a state for safety is proportional to $m^a n^b$
- What are the values of *a* and *b*?

Problem 6.25 – Solution

- Comparing a row in the matrix to the vector of available resources takes *m* operations. This step must be repeated on the order of *n* times to find a process that can finish and be marked as done
- Thus, marking a process as done takes on the order of *mn* steps. Repeating the algorithm for all *n* processes means that the number of steps is then $O(mn^2)$
- Thus, $a = 1$ and $b = 2$

Work in pairs
Think-pair-share

Problem 6.26

- A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows:

	Allocated	*Maximum*	*Available*
Process A	1 0 2 1 1	1 1 2 1 3	0 0 x 1 1
Process B	2 0 1 1 0	2 2 2 1 0	
Process C	1 1 0 1 0	2 1 3 1 0	
Process D	1 1 1 1 0	1 1 2 2 1	

- What is the smallest value of ***x*** for which this is a safe state?

Problem 6.26 – Solution (1/2)

- The needs matrix is:
  0 1 0 0 2
  0 2 1 0 0
  1 0 3 0 0
  0 0 1 1 1
- If *x* is 0, we have a deadlock immediately
- If *x* is 1, process D can run. When it is finished, the available vector is 1 1 2 2 1
  Now we are deadlocked

Problem 6.26 – Solution (2/2)

- If x is 2, after D runs, the available vector is 1 1 3 2 1 and C can run
- After it finishes and returns its resources the available vector is 2 2 3 3 1, which will allow B to run and complete, and then A to run and complete
- Therefore, the smallest value of x that avoids a deadlock is 2

Problem 6.34

- Explain the differences between
  - deadlock,
  - livelock, and
  - starvation (possibly, using one-lane bridge analogy)

Problem 6.34 – Solution (1/4)

- For all cases, bridge has an entry and an exit which can be considered as resources of different types. A car will cross the bridge if and only if it takes both resources
- Now, let's consider three cases: deadlock, livelock and starvation

Problem 6.34 – Solution (1/3)

- Deadlock:
  - Two cars are entering the bridge from different sides. Each car captures one resource. If both car drivers are stubborn, no one will willingly free a resource that other one needs
  - Consequently, we have a deadlock, since both cars will stuck forever

Problem 6.34 – Solution (1/3)

- Livelock:
  - Let's assume that both drivers are polite and they decide to get back and let other driver pass the bridge, than try entering the bridge again
  - If both follow the same strategy, they will stuck forever. However, they will continue entering and leaving the bridge, so this situation is not a deadlock since some activity still happens

Problem 6.34 – Solution (1/3)

- Starvation:
  - Consider the bridge with a sign that says that trucks have lower priority than passenger cars and can cross the bridge if and only if there are no such cars at both ends
  - In this case, theoretically, a truck might stuck for a long time (or forever) if passenger cars constantly arrive at the bridge

Problem 6.12

- In order to control traffic, a network router, A periodically sends a message to its neighbor, B, telling it to increase or decrease the number of packets that it can handle.
- At some point in time, Router A is flooded with traffic and sends B a message telling it to cease sending traffic.
- It does this by specifying that the number of bytes B may send (A's window size) is 0.
- As traffic surges decrease, A sends a new message, telling B to restart transmission. It does this by increasing the window size from 0 to a positive number. That message is lost. As described, neither side will ever transmit.
- What type of deadlock is this?

Problem 6.12 – Solution

- This is a communication deadlock, and can be controlled by having A time out and retransmit its enabling message (the one that increases the window size) after some period of time (a heuristic)
- It is possible, however, that B has received both the original and the duplicate message. No harm will occur if the update on the window size is given as an absolute value and not as a differential
- Sequence numbers on such messages are also effective to detect duplicates

End

Week 13 – Tutorial