C211 – Operating Systems

Tutorial: Deadlocks

– Answers –

1. Suppose that there is a resource deadlock in a system. Give an example to show that the set of processes deadlocked can include processes that are not in the circular chain in the corresponding resource allocation graph.

   Consider three processes A, B and C, and two resources R and S. Suppose A is waiting for R that is held by B, B is waiting for S held by A, and C is waiting for R held by B. All three processes, A, B and C are deadlocked. However, only A and B belong to the circular chain.

2. Consider a system that uses the banker's algorithm to avoid deadlocks. At some time a process P requests a resource R, but is denied even though R is currently available. Does it mean that if the system allocated R to P, the system would deadlock?

   No. An available resource is denied to a requesting process in a system using the banker's algorithm if there is a possibility that the system may deadlock by granting that request. It is certainly possible that the system may not have deadlocked if that request was granted.

3. Two processes, A and B, each need three records, `1`, `2`, and `3`, in a database. If A asks for them in the order `1`, `2`, `3`, and B asks for them in the same order, deadlock is not possible. However, if B asks for them in the order `3`, `2`, `1`, then deadlock is possible. With three resources, there are 3! = 6 possible combinations each process can request resources. What fraction of all combinations is guaranteed to be deadlock free?

   Answer: Suppose that process A requests the records in the order `a`, `b`, `c`. *If process B also asks for* `a` *first, one of them will get it and the other will block. This situation is always deadlock free since the winner can now run to completion without interference. The other four combinations can be similarly reasoned about and shown to lead to possible deadlock:*

   (1) `a b c`*: deadlock free*

   (2) `a c b`*: deadlock free*

   (3) `b a c`*: possible deadlock*

   (4) `b c a`*: possible deadlock*

   (5) `c a b`*: possible deadlock*

   (6) `c b a`*: possible deadlock*

   So only one third of the cases are guaranteed to be deadlock free.

4. Can a single-processor system have no processes ready and no process running? Is this a deadlocked system? Explain your answer.

In theory, we might have a single blocked process waiting for an I/O completion; when the completion occurs, the blocked process proceeds to the ready state (or even directly to the running state). However, in practice, most architectures require a valid process to execute at all times. Therefore, many operating systems assign an idle process created by the kernel to execute when no other processes are ready. Although this system may appear not to be doing anything, it is not dead in the sense that as soon as the I/O completion interrupt arrives, the system will indeed resume operation.