Deadlock Avoidance

This approach to the deadlock problem anticipates deadlock before it actually occurs. This approach employs an algorithm to access the possibility that deadlock could occur and acting accordingly. This method differs from deadlock prevention, which guarantees that deadlock cannot occur by denying one of the necessary conditions of deadlock.

If the necessary conditions for a deadlock are in place, it is still possible to avoid deadlock by being careful when resources are allocated. Perhaps the most famous deadlock avoidance algorithm, due to Dijkstra [1965], is the Banker's algorithm. So named because the process is analogous to that used by a banker in deciding if a loan can be safely made.

Banker's Algorithm

In this analogy

Customers	≡	processes
Units	≡	resources, say, tape drive
Banker	≡	Operating System

Customers	Used	Max	
A	0	6	
B	0	5	Available Units
C	0	4	= 10
D	0	7	

Fig. 1

In the above figure, we see four customers each of whom has been granted a number of credit nits. The banker reserved only 10 units rather than 22 units to service them. At certain moment, the situation becomes

Customers	Used	Max	
A	1	6	
B	1	5	Available Units
C	2	4	= 2
D	4	7	

Fig. 2

Safe State The key to a state being safe is that there is at least one way for all users to finish. In other analogy, the state of figure 2 is safe because with 2 units left, the banker can delay any request except *C's*, thus letting *C* finish and release all four resources. With four units in hand, the banker can let either *D* or *B* have the necessary units and so on.

Unsafe State Consider what would happen if a request from B for one more unit were granted in above figure 2.

We would have following situation

Customers	Used	Max	
A	1	6	
B	2	5	Available Units
C	2	4	= 1
D	4	7	

Fig. 3

This is an unsafe state.

If all the customers namely *A*, *B*, *C*, and *D* asked for their maximum loans, then banker could not satisfy any of them and we would have a deadlock.

Important Note: It is important to note that an unsafe state does not imply the existence or even the eventual existence a deadlock. What an unsafe state does imply is simply that some unfortunate sequence of events might lead to a deadlock.

The Banker's algorithm is thus to consider each request as it occurs, and see if granting it leads to a safe state. If it does, the request is granted, otherwise, it postponed until later. Haberman [1969] has shown that executing of the algorithm has complexity proportional to N^2 where N is the number of processes and since the algorithm is executed each time a resource request occurs, the overhead is significant.

Source:

http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/deadlockAvoidance.htm