CmpE 322

January 29, 2022

Contents

1 Discussion VII

1) Deadlock

(a) Can you think of a deadlock without circular wait? If yes, please give an example.

Answer: No, circular wait is one of the four necessary conditions for a deadlock situation to arise: Mutual exclusion, hold-and-wait, no preemption and circular wait.

(b) Does your answer change, if we have only two processes?

Answer: No, still circular wait is a necessity.

2) Unsafe States

(a) What is an unsafe state in the context of deadlocks?

Answer: Safe state is the state in which there exists a sequence $<P_1$, P_2, ..., $P_n>$ of all the processes in the systems such that for each P_i, the resources that P_i can still request can be satisfied by currently available resources + resources held by all P_j with j<i. If this condition is satisfied, then no deadlock is guaranteed. Because when P_i terminates, it is guaranteed that the P_{i+1} can obtain its needed resources, and so on. An unsafe state is a state that is not safe, therefore doesn't guarantee the allocation of the resources as in the safe state case.

(b) How can we have an unsafe state but no deadlock?

Answer: Unsafe state doesn't necessarily lead to a deadlock. Say, there are 2 processes A and B. Both of them requests resources X and Y, therefore state is unsafe. However, by chance there may not occur deadlock. Let's assume a context switch occurred from process A to process B after process A locked resources X and Y to utilize them. Since resources are locked, B cannot be allocated the requested resources. Then, assume another context switch

occurred from process B to A and A has finished its work with the resources, therefore it unlocked them. Now a context switch occurs and Y can continue with the resources. However it could be the case that A locked X and B locked Y, therefore they are waiting for each other, which leads to a deadlock.

3) Remember that a priori information about the possible use of the resources is required by the Deadlock Avoidance algorithms (e.g. Banker's Algorithm). Is this a realistic assumption? If not, why not? If so, how can we obtain a priori information?

Answer: It is a realistic assumption, a priori information can be obtained for the deadlock avoidance algorithms. Compiler can extract the information about a program during the compilation step. While creating the compiled executable, it can place the information so that Operating System receives the priori information about the program when it is placed to memory for execution.

4) Assume the states of a system initially and after a few resource allocations are as shown in the following table. To be able to avoid deadlocks by using Banker's Algorithm, identify the question-marked states as safe or unsafe, and also explain your reasoning. Assume there is only one type of resource, i.e. only buffers.

Answer: For the second, third and fourth cases, states are safe. However, in the fifth case, there cannot be total ordering and therefore state is unsafe.

5) If there are methods for deadlock avoidance and prevention, why do we still need mechanisms for deadlock detection instead of avoiding them?

Answer: Deadlock prevention strategies may cause inefficiencies in our system or it may not be suitable for some cases. Mutual exclusion condition must hold if at least one resource is non-shareable. Preventing hold-and-wait introduces two main disadvantages: low resource utilization and starvation of a process, because at least one of the resources that it needs is always allocated to some other process. Preemption protocol cannot be applied to resources such as mutex locks and semaphores, and it doesn't seem to be an efficient solution since we have to reallocate all resources again and again between the processes. Lastly, for the circular wait case, imposing a lock ordering does not guarantee deadlock prevention if locks can be acquired dynamically. Possible side effects of preventing deadlocks with deadlock-prevention algorithms are low device utilization and reduces system throughput. Overall, deadlock prevention may not always be the best solution and we can implement deadlock detection algorithms rather than using deadlock prevention protocols.

Deadlock avoidance algorithms require additional information about how resources are to be requested. In cases when information is not available, deadlock avoidance cannot be used. In reality, deadlocks rarely occur and therefore it's a good idea to develop deadlock detection models for the cases when neither of the deadlock prevention models are used.

6) Deadlock Victim

(a) What is a victim while trying to recover from a deadlock?

Answer: There are two options for breaking a deadlock. One is simply to abort one or more processes to break the circular wait. The other is to preempt

some resources from one or more of the deadlocked processes. In resource preemption, we have to select a victim process to be preempted.

(b) Which issues should be considered while choosing a victim for resolving a deadlock?

Answer: Order of preemption must be determined to minimize the cost. Cost factors may include such parameters as the number of resources a deadlocked process is holding and the amount of time the process has thus far consumed. Other issues are priority of the process, how many processes will need to be terminated so resolve the deadlock, whether the process is interactive or batch (batch processes are preferred for termination).

7) Deadlock in Multiprocessor Systems

(a) Can we have a deadlock in a multiprocessor system? Why or why not?

Answer: We can have a deadlock in a multiprocessor system because deadlock requires mutual exclusion and no preemption. Mutual exclusion guarantees that no other process can operate on their critical region while a process is already operation on its critical region. No preemption guarantees that a resource cannot be allocated to another process from a process by force. If we have such conditions, we can also encounter with hold-and-wait and circular wait, and therefore a deadlock in a multiprocessor system. The difference from a uniprocessor system is that it can run more than one process in parallel, however it doesn't make any change for the deadlocks since a deadlock can occur if mutual exclusion is provided.

(b) If the resources in the multiprocessor system are allocated in a preemptive manner, does your answer to the same question change? Why or why not?

Answer: No preemption is one of the requirements for a deadlock to occur. If resources are allocated in a preemptive manner, deadlock situations can be resolved by preempting a process and allocating the related resources to another process to continue the execution.