Operating system concepts

Process Synchronization (deadlocks detection, prevention, avoidance)

Slides Set #13

By Prof K R Chowdhary

JNV University

2023

Deadlock Prevention: No Preemption, Circular Wait

3. No Preemption. The third necessary condition for deadlocks to occur is: there be no preemption of resources that have already been allocated. To ensure that ...

- Alternatively, if a process requests some resources, we first check whether they are available. If they are, we allocate them. If they are not,...
- This protocol is often applied to resources whose state can be easily saved and restored later,
- Questions: 1. In what conditions, the preemption of resources cannot be applied? 2. Explain the preemption protocol of deadlock removal, in detail.

4. Circular Wait. The fourth and final condition for deadlocks is the circular-wait condition. To ensure that it does not hold is impose **total order** of resources.
We define a one-to-one function $f : R \rightarrow N$

Deadlock Avoidance

- * Deadlock-prevention algorithms ensures that **at least one of the necessary conditions for deadlock cannot occur**. Side effects?
- 1. A method for avoiding deadlocks is to require **additional information** for example, in a system with "one tape drive and one printer," the system...
- 2. A deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that a circular-wait condition can never exist.
- A state is safe if the system can allocate resources to each process (up to its maximum) in some order and still avoid a deadlock.
- Questions: 1. What is disadvantages of method * above? 2. What are the different possible mechanisms for deadlock avoidance?

Deadlock detection

- If a system does not employ either a deadlock-prevention or a deadlock- avoidance algorithm, then a deadlock situation may occur. In this environment, the system may provide:

1. An algorithm that examines the state of the system to determine whether a deadlock has occurred, and
2. An algorithm to recover from the deadlock.

Recovery from deadlock

When a detection algorithm determines that a deadlock exists, several alter- natives are available.

1. Simplest solution is is to inform the operator that a deadlock has occurred and to let the operator deal with the deadlock manually.
2. Another possibility is to let the system recover from the deadlock automatically.

- Two options for breaking a deadlock: 1. simply to abort one or more processes to break the circular wait. 2. preempt some resources from one or more of the deadlocked processes.
- **Abort all deadlocked processes.** This method clearly will break the deadlock cycle, but at great expense.
- Abort one process at a time until the deadlock cycle is eliminated.
- Questions: 1. "Abort all deadlocked processes" method has what disadvantages?

Recovery from deadlock: Resource preemption

If preemption is required to deal with deadlocks, then three issues need to be addressed:

- **Selecting a victim**. Which resources and which processes are to be preempted?
- **Rollback.** If we preempt a resource from a process, what should be done with that process?
- **Starvation.** How do we ensure that starvation will not occur? That is, how can we guarantee that resources will not always be preempted from the same process?