Deadlock Prevention

Deadlock Prevention

- **Attacking the Mutual Exclusion Condition**
- **Attacking the Hold and Wait Condition**
- **Attacking the No Preemption Condition**
- **Attacking the Circular Wait Condition**

Deadlock Prevention

Attacking the Mutual Exclusion Condition

- Mutual exclusion automatically holds for printers and other non- sharables.
- Shared entities (read only files) don't need mutual exclusion (and aren’t susceptible to deadlock.)
- Prevention is not possible, since some devices are intrinsically non-sharable.

Deadlock Prevention

Attacking the Hold and Wait Condition

- To ensure that the hold-and-wait condition never occurs in the system, whenever a process requests a resource, it does not hold any other resources.

Method I

- All processes request all their resources before starting execution.
- If everything is available, the process will be allocated whatever it needs and can run to completion.
- If one or more resources are busy, nothing will be allocated and the process would just wait.

Problem

- Many processes do not know how many resources they will need until they have started running.
- Resources will not be used optimally with this approach.

Deadlock Prevention

Method-II

- A process requesting a resource to first temporarily release all the resources it currently holds.
- Then it tries to get everything it needs all at once

Attacking the No Preemption Condition

Method-I

- If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources the process is currently holding are preempted.
- The preempted resources are added to the list of resources for which the process is waiting. The process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting.

Deadlock Prevention

Method-II

- If a process requests some resources, first check whether they are available.
- If they are, allocate them.
- If they are not, check whether they are allocated to some other process that is waiting for additional resources.
- If so, preempt the desired resources from the waiting process and allocate them to the requesting process.
- If the resources are neither available nor held by a waiting process, the requesting process must wait.
- While it is waiting, some of its resources may be preempted, but only if another process requests them.
- A process can be restarted only when it is allocated the new resources it is requesting and recovers any resources that were preempted while it was waiting

Deadlock Prevention

Attacking the Circular Wait Condition

- The circular wait can be eliminated in several ways.

Method –I

- Simply to have a rule saying that a process is entitled only to a single resource at any moment.
- If it needs a second one, it must release the first one.

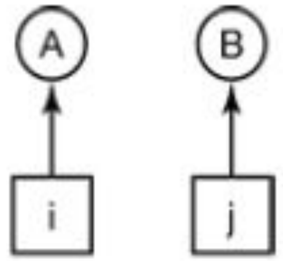
Method-II

- Provide a global numbering of all the resources.
- **Rule:** processes can request resources whenever they want to, but all requests must be made in numerical order.

Deadlock Prevention

1. Imagesetter
2. Scanner
3. Plotter
4. Tape drive
5. CD Rom drive

(a)



(b)

Numerically ordered resources. (b) A resource graph.

- A process may request first a scanner and then a tape drive, but it may not request first a plotter and then a scanner.

Deadlock Prevention

- With this rule, the resource allocation graph can never have cycles.
- Deadlock will occur only if *A requests resource j and B requests resource i.*
- *Assuming i and j are distinct* resources, they will have different numbers.
- If *i > j, then A is not allowed to request j* because that is lower than what it already has.
- If *i < j, then B is not allowed to request i* because that is lower than what it already has.
- Either way, deadlock is impossible.