DEADLOCKS

CS646: Operating System

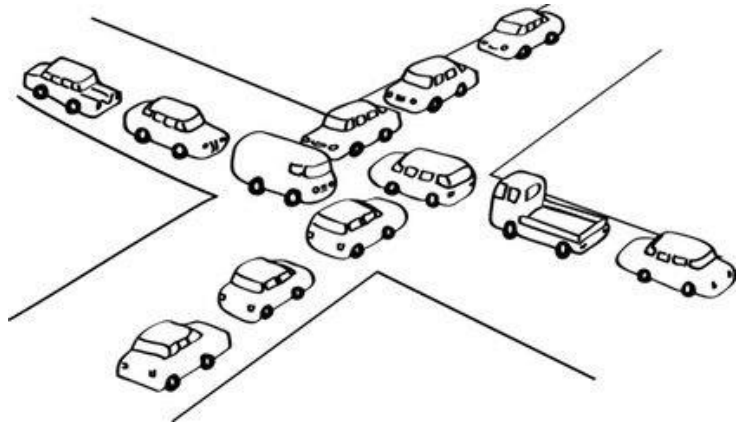
Mohammad Farhan

Outcomes

- Learn about Deadlocks, which prevent sets of concurrent processes from completing their tasks.
- Learn about the reasons of Deadlocks.
- Learn about several different methods for preventing or avoiding deadlocks in a computer system.

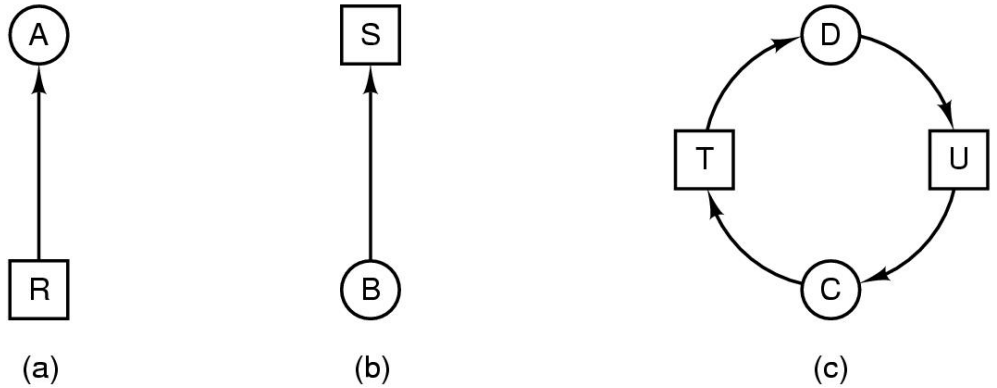
Deadlocks

- A condition where two or more threads are waiting for an event that can only be generated by these same threads.
- It occurs when two processes attempt to gain exclusive access to a resource, and each of them waits on the other to complete before moving forward.
- Deadlocks can occur via system calls, locking, etc.



Example

- Process uses resources following way
  - Requests a resource
  - Use the resource
  - Releases the resource
- Modeled with directed graphs
  - resource R assigned to process A.
  - process B is requesting/waiting for resource S.
  - process C and D are in deadlock over resources T and U.
  - If the graph has no cycles, no deadlock exists.
  - If the graph has cycle, deadlock might exist.



Example

Request R
Request S
Release R
Release S

(a)

Request S
Request T
Release S
Release T

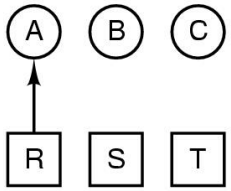
(b)

Request T
Request R
Release T
Release R

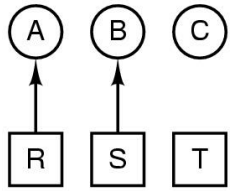
(c)

1. A requests R
2. B requests S
3. C requests T
4. A requests S
5. B requests T
6. C requests R
   deadlock

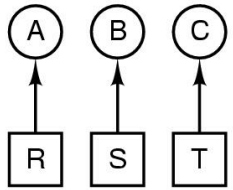
(d)



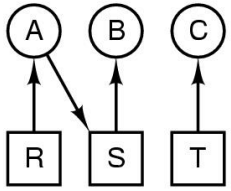
(e)



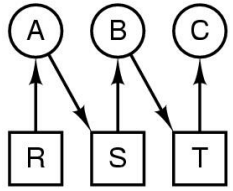
(f)



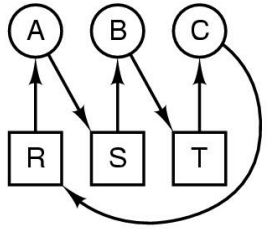
(g)



(h)



(i)



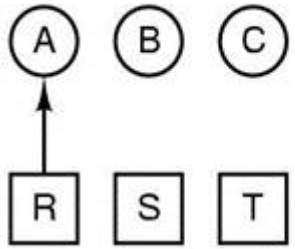
(j)

Example

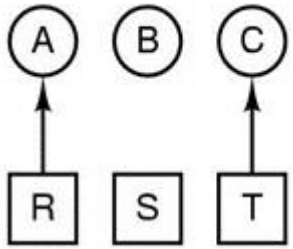
1. A requests R
2. C requests T
3. A requests S
4. C requests R
5. A releases R
6. A releases S

no deadlock

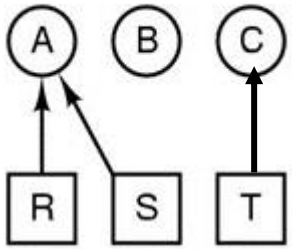
(k)



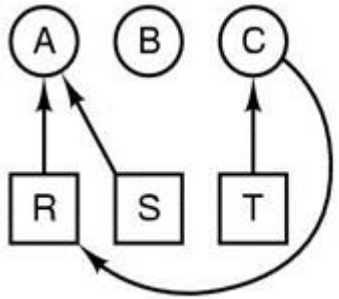
(l)



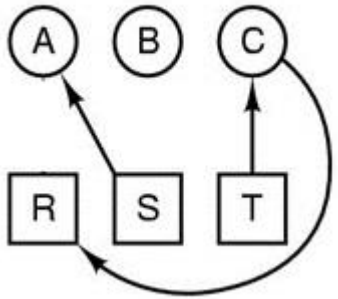
(m)



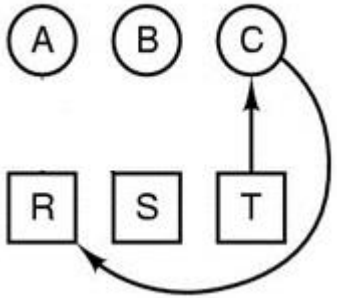
(n)



(o)



(p)



(q)

Conditions for Deadlock

- Mutual Exclusion: at least one thread must hold a resource in non sharable mode, i.e., the resource may only be used by one thread at a time.
- Hold and Wait: at least one thread holds a resource and is waiting for other resource(s) to become available. A different thread holds the resource(s).
- No Preemption: A thread can only release a resource voluntarily; another thread or the OS cannot force the thread to release the resource.
- Circular Wait: A set of waiting threads $\{t_1, ..., t_n\}$ where t_i is waiting on t_{i+1} (i = 1 to n) and t_n is waiting on t_1.

Deadlock Prevention

- Prevent deadlock: ensure that at least one of the necessary conditions doesn't hold.

1. Mutual Exclusion: make resources sharable (but not all resources can be shared).
2. Hold and Wait:
   - Guarantee that a thread cannot hold one resource when it requests another.
   - Make threads request all the resources they need at once
   - Make the thread release all resources before requesting a new set.
3. No Preemption:
   - If a thread requests a resource that cannot be immediately allocated to it, then the OS preempts (releases) all the resources that the thread is currently holding.
   - Only when all the resources are available, will the OS restart the thread.
4. Circular Wait: impose an ordering (numbering) on the resources and request them in order.

Summary

- Deadlock: situation in which a set of threads/processes cannot proceed because each requires resources held by another member of the set.
- Detection and Recovery: recognize deadlock after it has occurred and break it.
- Avoidance: don't allocate a resource if it would introduce a cycle.
- Prevention: design resource allocation strategies that guarantee that one of the necessary conditions never holds.

THANK YOU