Chapter 6

Deadlock

(A Quick Introduction)

Resources and Deadlocks

Processes need access to resources in order to make progress.

Examples of Resources:

- **Kernel Data Structures**
  (ProcessControlBlocks, Threads, OpenFile…)
- **Locks/semaphores to protect critical sections**
- **Memory (page frames, buffers, etc.)**
- **Files**
- **I/O Devices**
  (printers, ports, tape drives, speaker, etc.)

Resources and Deadlocks

Scenario:

Process P1...
is holding resource A, and
is requesting resource B

Process P2...
is holding resource B, and
is requesting resource A

Both are blocked and remain so …

This is deadlock

Resource Usage Model

Sequence of events required to use a resource:

- **request the resource (e.g., acquire a mutex lock)**
- **use the resource**
- **release the resource (e.g., release a mutex lock)**

Must wait if request is denied

Preemptable vs Nonpreemptable Resources

Preemptable resources

Can be taken away from a process with no ill effects

Nonpreemptable resources

Once given to the process, can't be taken back

"Deadlocks occur when processes are granted exclusive access to non-preemptable resources and wait when the resource is not available."

Definition of Deadlock

"A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause."

Usually the event is:

The release of a currently held resource

All processes in the set are waiting

... for a resource request to be granted.

None of the processes can proceed

... so no process can release the resources it holds.

Starvation vs. Deadlock

Starvation **and** ***Deadlock*** **are two different things!**

Deadlock:

- **No work is being accomplished for the processes that are deadlocked, because processes are waiting for each other. Once present, will not go away!**

Starvation:

- **Work (progress) is occurring. However, a particular set of processes may not be getting any work done because they cannot obtain the resources they need.**
- **May only last a short time; may go away.**

Both are probabilistic events & may occur only rarely.

Deadlock Conditions

A deadlock situation can occur *if and only if* the following conditions hold simultaneously...

Mutual Exclusion Condition

A resource can be assigned to only one process at a time

Hold And Wait Condition

Processes can get more than one resource

No Preemption Condition

Circular Wait Condition

A cyclic chain of two or more processes (must be waiting for resource from next one in chain)

Resource acquisition scenarios

Thread A:

```
acquire (resource_1)
use resource_1
release (resource_1)
```

Example:

```
var r1_mutex: Mutex
...
r1_mutex.Lock()
Use resource_1
r1_mutex.Unlock()
```

Resource acquisition scenarios

Thread A:

```
acquire (resource_1)
use resource_1
release (resource_1)
```

Another Example:

```
var r1_sem: Semaphore
r1_sem.Up()
...
r1_sem.Down()
Use resource_1
r1_sem.Up()
```

Resource acquisition scenarios

Thread A:

```
acquire (resource_1)
use resource_1
release (resource_1)
```

Thread B:

```
acquire (resource_2)
use resource_2
release (resource_2)
```

No deadlock can occur here!

Resource Acquisition Scenarios: 2 Resources

Thread A:

```
acquire (resource_1)
acquire (resource_2)
use resources 1 & 2
release (resource_2)
release (resource_1)
```

Thread B:

```
acquire (resource_1)
acquire (resource_2)
use resources 1 & 2
release (resource_2)
release (resource_1)
```

No deadlock can occur here!

Resource Acquisition Scenarios: 2 Resources

Thread A:

```
acquire (resource_1)
use resources 1
release (resource_1)
acquire (resource_2)
use resources 2
release (resource_2)
```

Thread B:

```
acquire (resource_2)
use resources 2
release (resource_2)
acquire (resource_1)
use resources 1
release (resource_1)
```

No deadlock can occur here!

Resource Acquisition Scenarios: 2 Resources

Thread A:

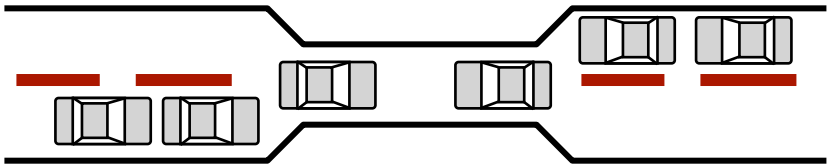
```
acquire (resource_1)
acquire (resource_2)
use resources 1 & 2
release (resource_2)
release (resource_1)
```

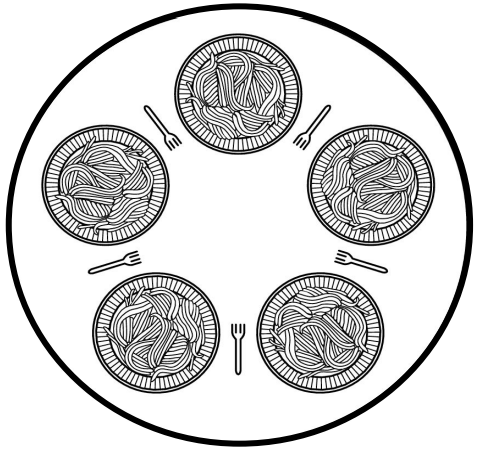
Thread B:

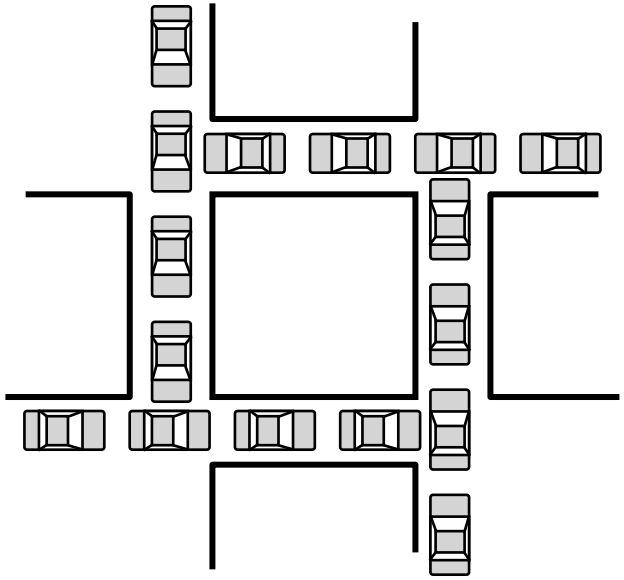
```
acquire (resource_2)
acquire (resource_1)
use resources 1 & 2
release (resource_1)
release (resource_2)
```

Deadlock is possible!

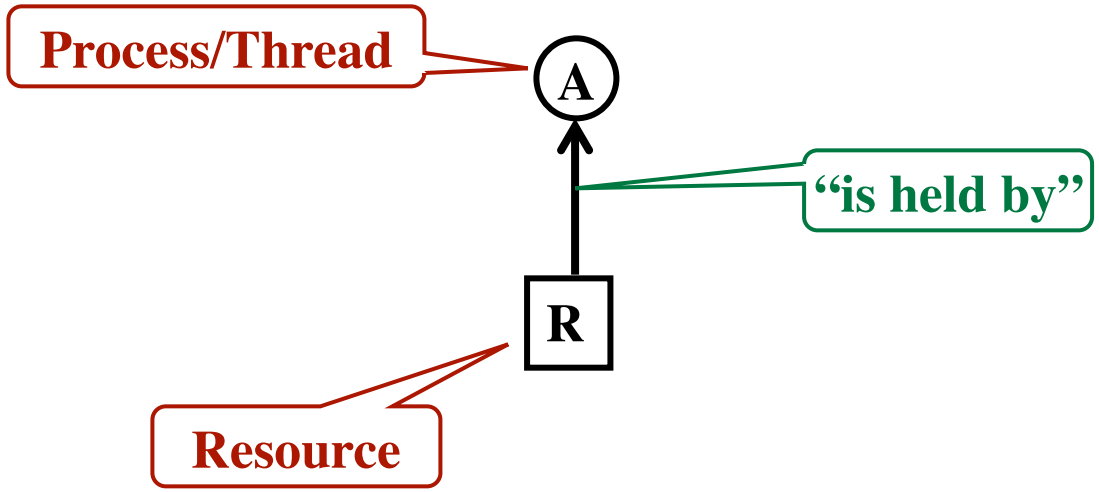
Other examples of deadlock



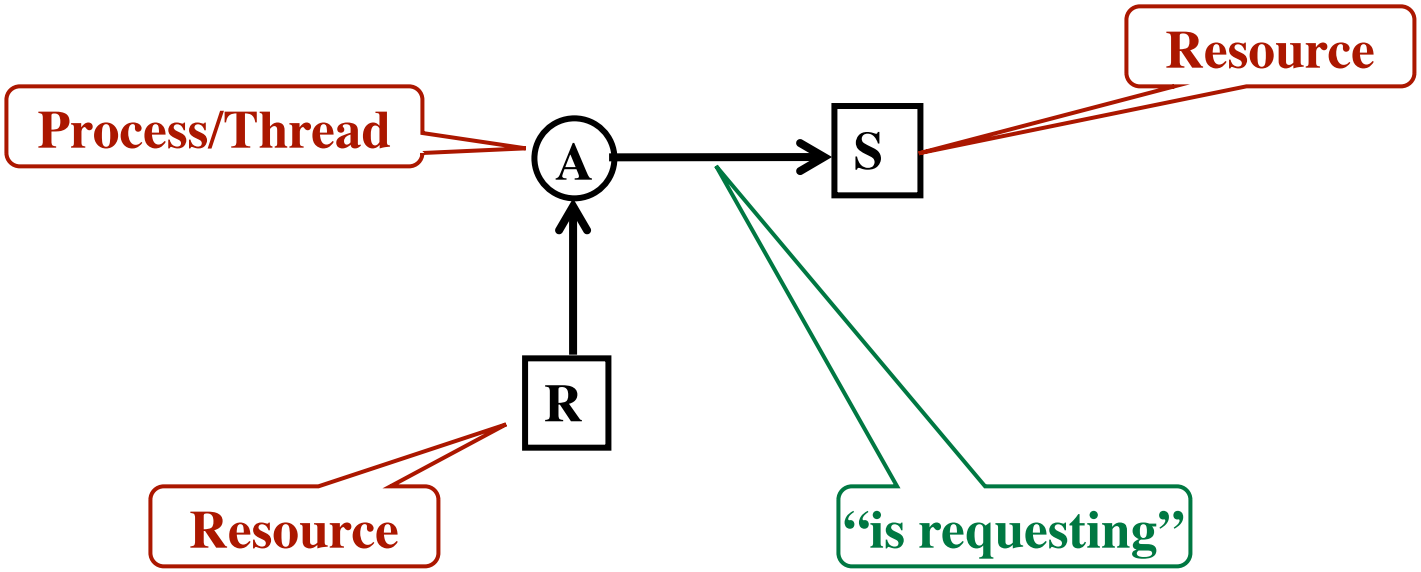




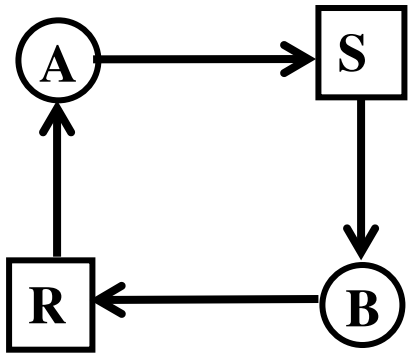
Resource Allocation Graphs



Resource Allocation Graphs



Resource Allocation Graphs



Deadlock = a cycle in the graph

Dealing with deadlock

General strategies

Ignore the Problem

Hmm… advantages, disadvantages?

Detection and Recovery

Avoidance

through careful resource allocation

Prevention

by structurally negating one of the four conditions

Recovery from Deadlock

What should be done to recover?

- **Abort deadlocked processes and reclaim resources**
- **Temporarily reclaim resource, if possible**
- **Abort one process at a time until deadlock cycle is eliminated**