

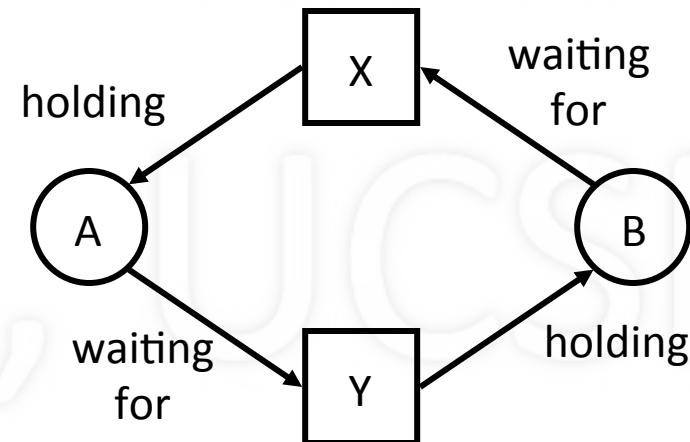
CSE 120: Principles of Operating Systems

Lecture 7: Deadlock

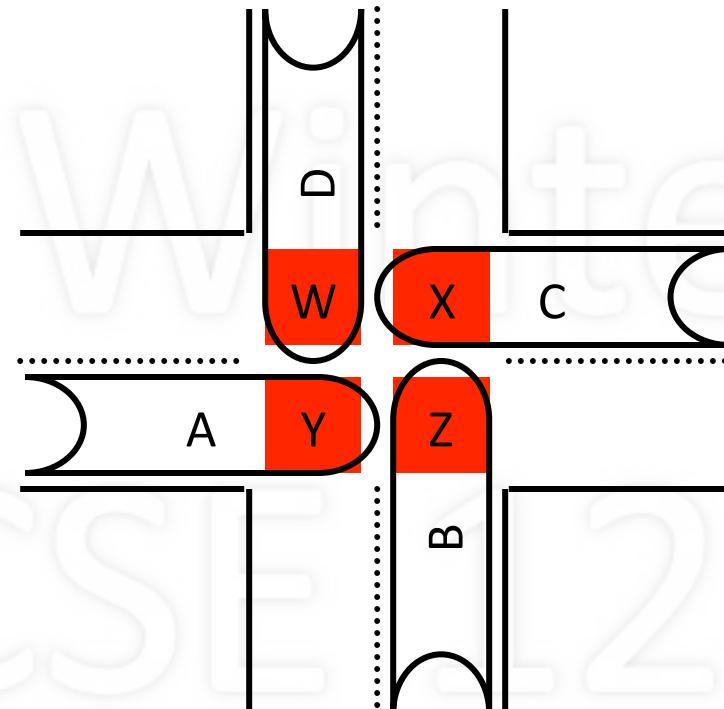
Prof. Joseph Pasquale
University of California, San Diego
January 30, 2019

What is Deadlock?

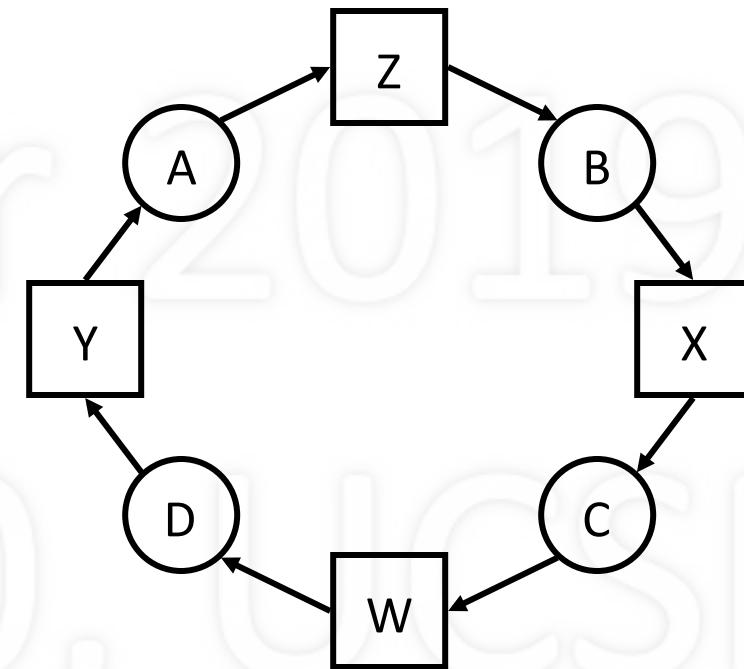
- Set of processes are permanently blocked
 - Unblocking of one relies on progress of another
 - But none can make progress!
- Example
 - Processes A and B
 - Resources X and Y
 - A holding X, waiting for Y
 - B holding Y, waiting for X
 - Each is waiting for the other; will wait forever



Traffic Jam as Example of Deadlock



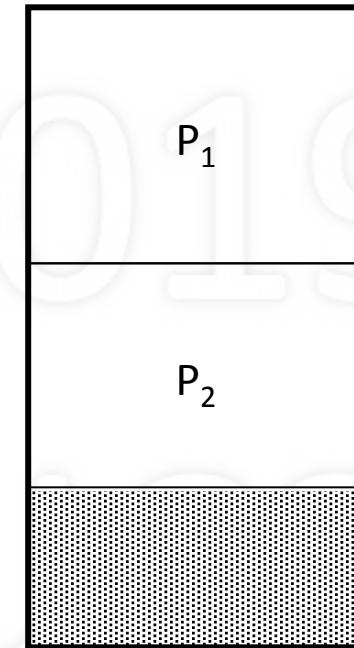
Cars deadlocked
in an intersection



Resource Allocation
Graph

Example of Deadlock: Memory

- Total memory = 200MB
- P_1 requests 80MB
- P_2 requests 70MB
- P_1 requests 60MB (wait)
- P_2 requests 80MB (wait)



Four Conditions for Deadlock

- Mutual Exclusion
 - Only one process may use a resource at a time
- Hold-and-Wait
 - Process holds resource while waiting for another
- No Preemption
 - Can't take a resource away from a process
- Circular Wait
 - The waiting processes form a cycle

How to Attack the Deadlock Problem

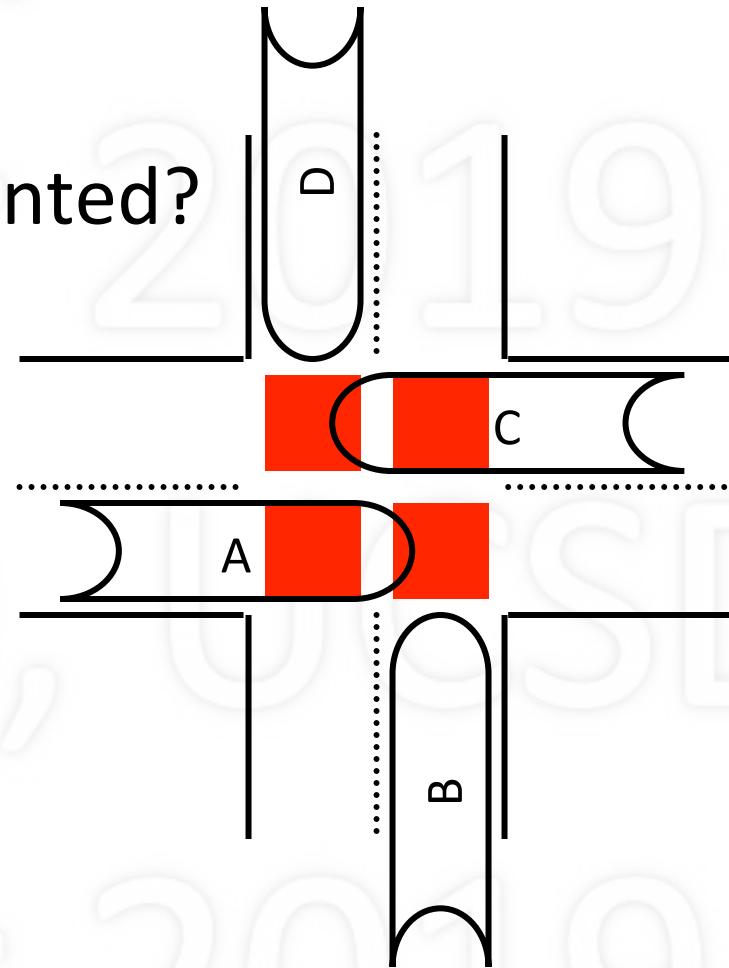
- Deadlock Prevention
 - Make deadlock impossible by removing condition
- Deadlock Avoidance
 - Avoid getting into situations that lead to deadlock
- Deadlock Detection
 - Don't try to stop deadlocks
 - Rather, if they happen, detect and resolve

Deadlock Prevention

- Simply prevent any single condition for deadlock
- Mutual exclusion
 - Relax where sharing is possible
- Hold-and-wait
 - Get all resources simultaneously (wait until all free)
- No preemption
 - Allow resources to be taken away
- Circular wait
 - Order all the resources, force ordered acquisition

How Can We Prevent a Traffic Jam?

- Add a traffic light
- Which condition is prevented?
 - Mutual exclusion?
 - Hold-and-wait?
 - No Preemption?
 - Circular Wait?



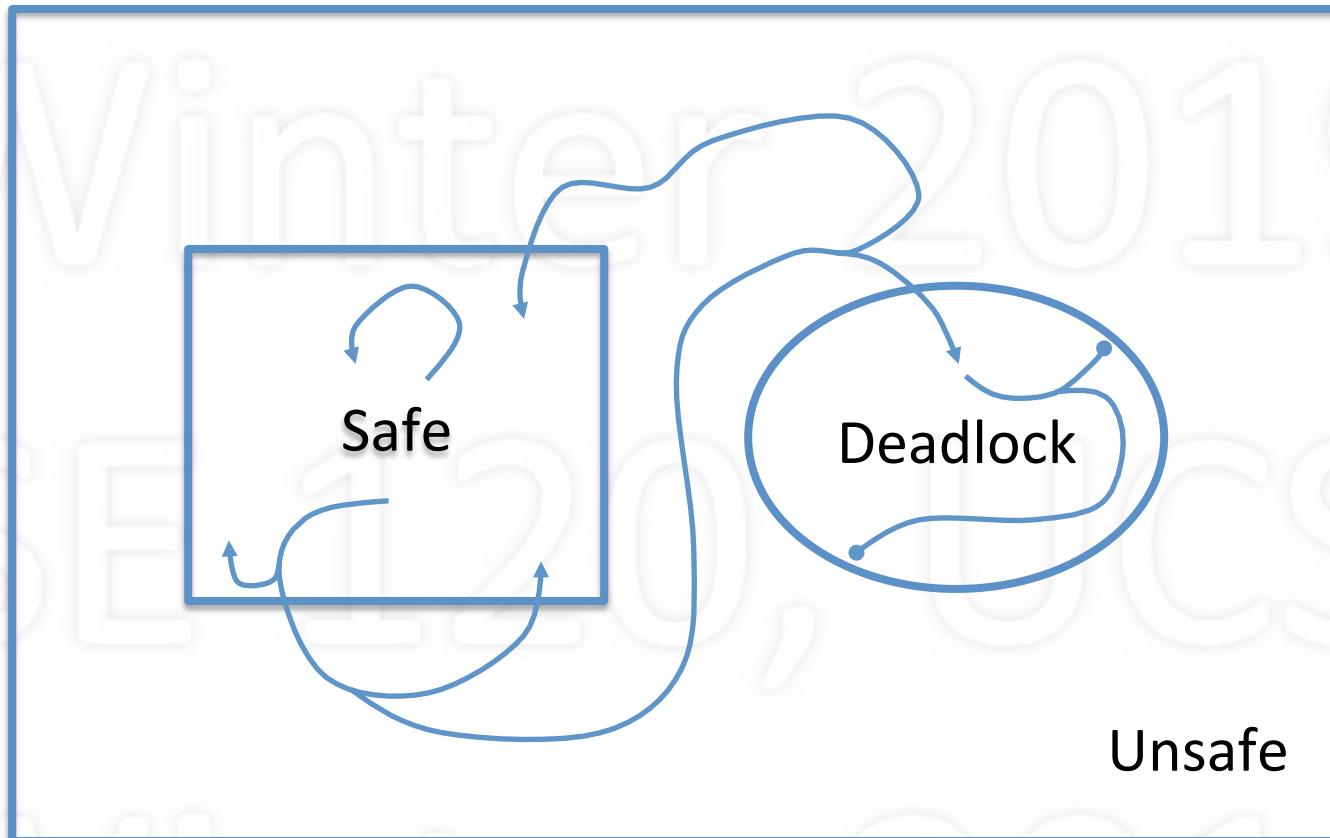
Deadlock Avoidance

- Avoid situations that lead to deadlock
 - Selective prevention
 - Remove condition only when deadlock is possible
- Works with incremental resource requests
 - Resources are asked for in increments
 - Do not grant request that can lead to a deadlock
- Need maximum resource requirements

Banker's Algorithm

- Fixed number of processes and resources
 - Each process has zero or more resources allocated
- System state: either safe or unsafe
 - Depends on allocation of resources to processes
- Safe: deadlock is absolutely avoidable
 - Can avoid deadlock by certain order of execution
- Unsafe: deadlock is possible (but not certain)
 - May not be able to avoid deadlock

Safe, Unsafe, and Deadlock States



Banker's Algorithm

- Given
 - process/resource claim matrix
 - process/resource allocation matrix
 - resource availability vector
- Is there a process ordering such that
 - a process can run to completion, return resources
 - resources can then be used by another process
 - eventually, all the processes complete

Example of a Safe State

	Claim				Allocation				Avail-ability	Total
	P ₁	P ₂	P ₃	P ₄	P ₁	P ₂	P ₃	P ₄		
R ₁	3	6	3	4	1	6	2	0	0	9
R ₂	2	1	1	2	0	1	1	0	1	3
R ₃	2	3	4	2	0	2	1	2	1	6

- This is a safe state
- Which process can run to completion? P₂
- After P₂ completes, its resources are returned
- Next select P₁, then P₃, then P₄

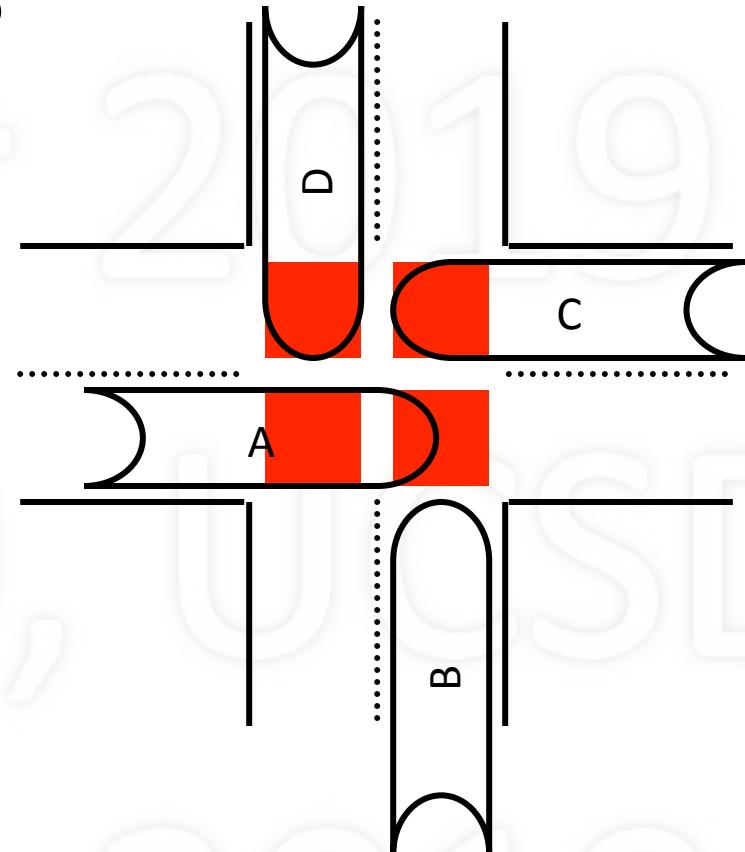
Example of an Unsafe State

	Claim				Allocation				Avail-ability	Total
	P ₁	P ₂	P ₃	P ₄	P ₁	P ₂	P ₃	P ₄		
R ₁	3	6	3	4	2	5	2	0	0	9
R ₂	2	1	1	2	0	1	1	0	1	3
R ₃	2	3	4	2	1	1	1	2	1	6

- This is an unsafe state
- No process can definitely run to completion
- P₁ may block asking for R₁; same for P₂, P₃, P₄
- Deadlock possible, but not necessarily certain

How Can We Prevent a Traffic Jam?

- What are the resources?
- What is a safe state?
- How to avoid deadlock?
 - At most 3 cars into intersection
- Which condition being prevented?

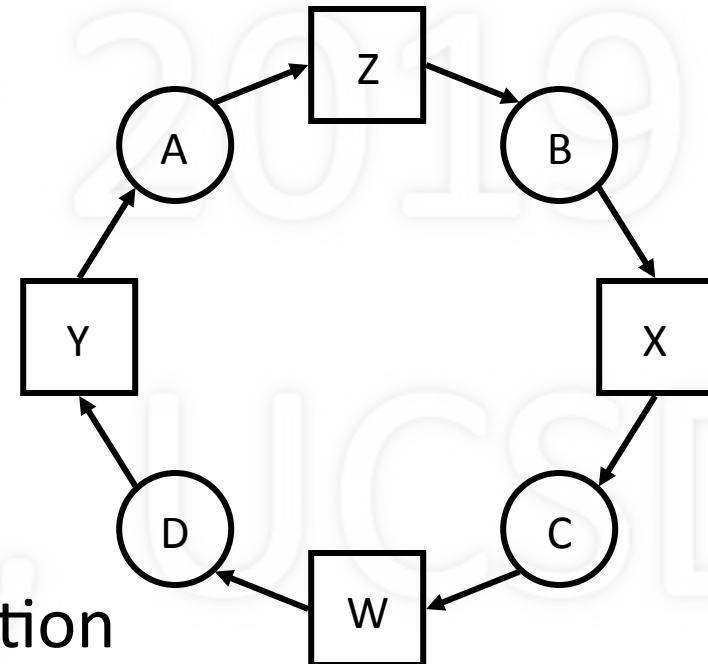


Deadlock Detection and Recovery

- Do nothing special to prevent/avoid deadlocks
 - If they happen, they happen
 - Periodically, try to detect if a deadlock occurred
 - Do something (or even nothing) about it
- Reasoning
 - Deadlocks rarely happen
 - Cost of prevention or avoidance not worth it
 - Deal with them in special way (may be very costly)
- Most general purpose OS's take this approach!

Detecting a Deadlock

- Construct resource allocation “wait-for” graph
 - If cycle, deadlock
- Requires
 - Identifying all resources
 - Tracking their use
 - Periodically running detection algorithm



Recovery from Deadlock

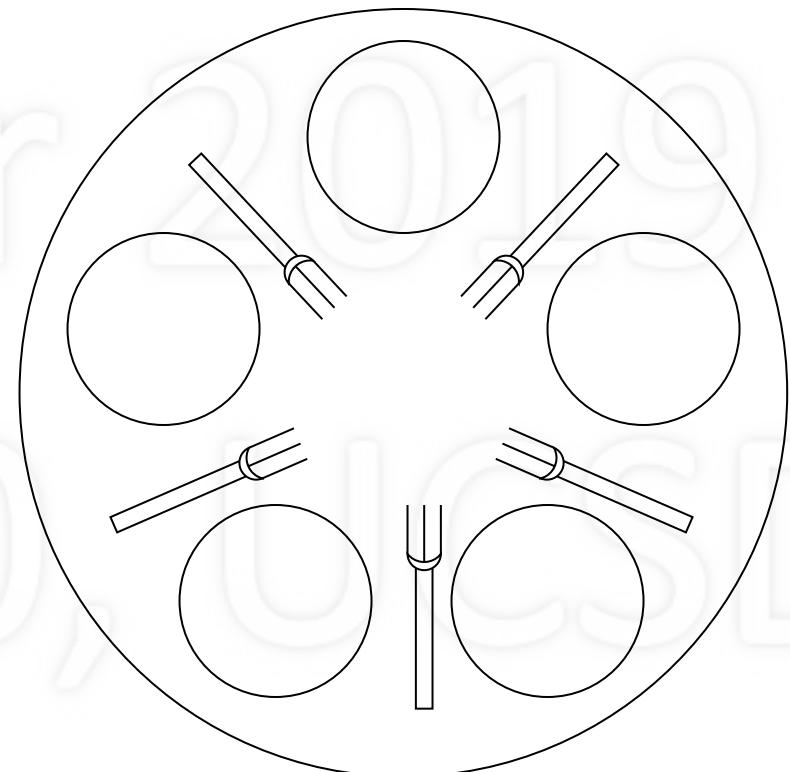
- Terminate all deadlocked processes
 - Will remove deadlock, but drastic and costly
- Terminate deadlocked processes one at a time
 - Do until deadlock goes away (need to detect)
 - What order should processes be ended?
- What about resources in inconsistent states
 - Such as files that are partially written?
 - Or interrupted message (e.g., file) transfers?

Classical Synchronization Problems

- The Producer/Consumer (Bounded Buffer) Problem
- The Dining Philosophers Problem
- The Readers/Writers Problem
- Study these problems and their solutions!

The Dining Philosopher's Problem

- Five philosophers
 - Think, eat, think, eat, ...
- To eat
 - Pick up two forks
 - one at a time
 - Eat
 - Put down forks
- Mutual exclusion
 - Avoid deadlock or starvation



Implementing Dining Philosophers

- Identify critical section(s)
- How to achieve mutual exclusion?
- How to avoid deadlock?
- How to avoid starvation?
- How to generalize to n philosophers?

```
DoPhilosopher (int i) {  
    while (TRUE) {  
        Think ();  
        PickupFork (i);  
        PickupFork ((i+1)%5);  
        Eat ();  
        PutdownFork ((i+1)%5);  
        PutdownFork (i);  
    }  
}
```

The Readers-Writers Problem

- Readers: processes that only read files
- Writers: processes that modify files
- Rules
 - Allow multiple simultaneous readers
 - A writer gets exclusive access
 - Avoid starvation: Once a writer arrives, wait until current readers leave, and then do not allow any new readers while writing

Textbook

- Chapter 8 (on Deadlocks)
 - Lecture-related: 8.1-8.9 (Deadlocks), 7.1 (Classic Synchronization Problems)
 - Recommended: 7.2-7.6

Review & Research

- What is meant by “deadlock”?
- When discussing deadlocks, what is meant by a “resource”?*
- In slide 3, what are the processes and what are the resources?*
- In slide 4, what are the processes and what are the resources?*

R&R

- What are the four conditions of deadlock?
- What is the “mutual exclusion” condition?
- What is the “hold-and-wait” condition?
- What is the “no preemption” condition?
- What is the “circular wait” condition?
- Are the conditions independent from each other; why or why not?**

R&R

- What is meant by Deadlock Prevention?
- What is meant by Deadlock Avoidance?
- What is meant by Deadlock Detection?
- How is Deadlock Prevention different from Deadlock Avoidance, and can you give an example?**
- Which is the easiest to implement; which is the hardest?**

R&R

- In Deadlock Prevention, for each of the four conditions for deadlock, how can they be prevented, and can you give an example of each in a real situation?**
- In slide 8, which condition is being prevented, and why?*
- What kind of operating system would use this technique?***

R&R

- In Deadlock Avoidance, why is it considered “selective prevention”?*
- What is meant by incremental resource requests?
- What is meant by “need maximum resource requirements”; can you give an example?*
- What kind of operating system would use this technique?***

R&R

- In the Banker's Algorithm, what is meant by a “safe state”?* What about “unsafe state”?*
 - In slide 11, what is being conveyed by the illustration?*
 - What is a process/resource claim matrix?*
 - What is a process/resource allocation matrix?
- *
- What is a resource availability vector?*

R&R

- What is the process ordering test used by the Banker's algorithm?*
- In slide 13, why is the state safe?*
- In slide 14, why is the state unsafe?*
- In slide 15, which condition is being avoided?
**

R&R

- What is the key justification for allowing deadlocks to happen in Deadlock Detection and Recovery?
- Why do most general-purpose operating systems use this technique?**
- What would be an example of an operating system that cannot use this technique?***

R&R

- In slide 17, what is the purpose of the wait-for graph?*
- How can an OS recover from deadlock?*
- Why is terminating all deadlocked processes costly?**
- What is the reasoning behind terminating processes one at a time, and what are some of the complications?**

R&R

- What is the Producer/Consumer (Bounded Buffer) Problem
- What is the Dining Philosophers Problem
- What is the Readers/Writers Problem

R&R

- In the Dining Philosopher's Problem, why is mutual exclusion needed?*
- In slide 21, can you identify the critical section(s)?**
- How is mutual exclusion achieved?**
- How is deadlock avoided?**
- How is starvation avoided?**
- How can you generalize to n philosophers?***

R&R

- How are readers and writers treated differently in the Readers-Writers Problem, and why?**
- Can you implement a solution to the Readers-Writers Problem?*** and one that avoids starvation?***