

CS 370: OPERATING SYSTEMS

[DEADLOCKS]

Shrideep Pallickara
Computer Science
Colorado State University

October 16, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.1

Frequently asked questions from the previous class survey

- Difference between request and assignment edges is simply the direction? Yes.
- Is D involved in a deadlock? $A \rightarrow B \rightarrow C \rightarrow A$ and $D \rightarrow A$
- Swap space?

October 16, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.2

Topics covered in this lecture

- Deadlock Prevention
- Deadlock Avoidance

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.3

DEADLOCK CHARACTERIZATION

October 16, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.4

Deadlocks: Necessary Conditions (I)

□ Mutual Exclusion

- At least one resource held in *nonsharable* mode
- When a resource is being used
 - Another requesting process must wait for its release

□ Hold-and-wait

- A process must hold one resource
- Wait to acquire additional resources
 - Which are currently held by other processes

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.5

Deadlocks: Necessary Conditions (II)

□ No preemption

- Resources cannot be preempted
- Only voluntary release by process holding it

□ Circular wait

- A set of $\{P_0, P_1, \dots, P_n\}$ waiting processes must exist
 - $P_0 \rightarrow P_1; P_1 \rightarrow P_2, \dots, P_n \rightarrow P_0$
- Implies hold-and-wait

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.6

DEADLOCK PREVENTION

October 16, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.7

Deadlock Prevention

- Ensure that **one** of the necessary conditions for deadlocks **cannot** occur
 - ① Mutual exclusion
 - ② Hold and wait
 - ③ No preemption
 - ④ Circular wait

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.8

Mutual exclusion must hold for non-sharable resources, but ...

- Sharable resources do not require mutually exclusive access
 - *Cannot be involved* in a deadlock
- A process never needs to wait for sharable resource
 - Read-only files
- Some resources are *intrinsically nonsharable*
 - So denying mutual exclusion often not possible

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.9

Deadlock Prevention: Ensure hold-and-wait never occurs in the system [Strategy 1]

- Process must request and be allocated all its resources **before** execution
 - Resource requests must precede other system calls
- E.g. copy data from DVD drive, sort file & print
 - Printer needed only at the end
 - BUT process will hold printer for the *entire* execution

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.10

Deadlock Prevention: Ensure hold-and-wait never occurs in the system [Strategy 2]

- Allow a process to request resources *only when it has none*
 - *Release* all resources, *before requesting* additional ones
- E.g. copy data from DVD drive, store file, & print
 - First request DVD and disk file
 - Copy and release resources
 - Then request file and printer

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.11

Disadvantages of protocols doing hold-and-wait

- **Low resource utilization**
 - Resources are allocated but unused for long durations
- **Starvation**
 - If a process needs several popular resources
 - Popular resource might always be *allocated to some other* process

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.12

Deadlock Prevention: Eliminate the preemption constraint

[1/2]

- {C1} If a process is holding some resources
- {C2} Process requests another resource
 - Cannot be immediately allocated

- All resources currently held by process is **preempted**
 - Preempted resources added to list of resources process is waiting for

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.13

Deadlock Prevention: Eliminate the preemption constraint

[2/2]

- Process requests resources that are not currently available
 - If resources allocated to another waiting process
 - Preempt resources from the second process and assign it to the first one

- Often applied when resource state can be **saved and restored**
 - CPU registers and memory space
 - Unsuitable for tape drives

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.14

Deadlock Prevention: Eliminating Circular wait

- Impose **total ordering** of all resource types
 - Assign each resource type a unique number
 - One-to-one function $F: R \rightarrow N$
 - $F(\text{tape drive}) = 1;$
 - $F(\text{printer}) = 12$
- ① Request resources in **increasing order**
- ② If several instances of a resource type needed?
 - Single request for all them must be issued

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.15

Requesting resources in an increasing order of enumeration

- Process initially requested R_i
- This process can now request R_j ONLY IF
$$F(R_j) > F(R_i)$$
- Alternatively, process requesting R_j must have released resources R_i such that
$$F(R_i) \geq F(R_j)$$
- Eliminates circular wait

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.16

Hierarchy of resources and deadlock prevention

- Hierarchy by itself does not prevent deadlocks
 - Developed programs **must follow ordering**

- **F** based on **order of usage** of resources
 - Tape drive needed before printing
 - $F(\text{tape drive}) < F(\text{printer})$

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.17

Deadlock Prevention: Summary

- Prevent deadlocks by **restraining** how requests are made
 - Ensure at least 1 of the 4 conditions **cannot** occur

- Side effects:
 - Low device utilization
 - Reduced system throughput

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.18

Dining Philosophers: Deadlock prevention (1)

- Mutual exclusion
 - Philosophers can *share* a chopstick

- Hold-and-wait
 - Philosopher should release the first chopstick if it cannot obtain the second one

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.19

Dining Philosophers: Deadlock prevention (2)

- Preemption
 - Philosophers can *forcibly take* each other's chopstick

- Circular-wait
 - Number the chopsticks
 - Pick up chopsticks in ascending order
 - Pick the lower numbered one before the higher numbered one

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.20

DEADLOCK AVOIDANCE

October 16, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.21

Deadlock avoidance

- Require *additional* information about **how** resources are to be requested
- Knowledge about sequence of requests and releases for processes
 - ▣ Allows us to decide if resource allocation *could cause a future deadlock*
 - ▣ Process P: Tape drive, then printer
 - ▣ Process Q: Printer, then tape drive

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.22

Deadlock avoidance: Handling resource requests

- For each resource request:
 - Decide whether or not process should wait
 - To avoid possible **future** deadlock

- Predicated on:
 - ① Currently available resources
 - ② Currently allocated resources
 - ③ Future requests and releases of each process

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.23

Avoidance algorithms differ in the amount and type of information needed

- **Resource allocation state**
 - Number of available and allocated resources
 - Maximum demands of processes

- Dynamically **examine** resource allocation state
 - Ensure circular-wait cannot exist

- Simplest model:
 - Declare maximum number of resources for each type
 - Use information to avoid deadlock

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.24

Safe sequence

- **Sequence** of processes $\langle P_1, P_2, \dots, P_n \rangle$ for the current allocation state
- Resource requests made by P_i can be satisfied by:
 - Currently available resources
 - Resources held by P_j where $j < i$
 - If needed resources not available, P_i can wait
 - In general, when P_i terminates, P_{i+1} can obtain its needed resources
- If no such sequence exists: system state is **unsafe**

October 16, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.25

Deadlock avoidance: Safe states

- If the system can:
 - ① Allocate resources to each process in **some order**
 - Up to the **maximum** for the process
 - ② Still avoid deadlock

October 16, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.26

Safe states and deadlocks

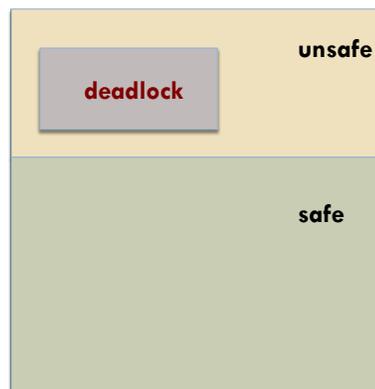
- A system is safe ONLY IF there is a **safe sequence**
- A safe state is not a deadlocked state
 - ▣ Deadlocked state is an unsafe state
 - ▣ Not all unsafe states are deadlocks

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.27

State spaces



October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.28

Unsafe states

- A unsafe state *may lead* to deadlock
- **Behavior** of processes controls unsafe states
- Cannot prevent processes from requesting resources such that deadlocks occur

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.29

Example: 12 Tape drives available in the system

	Maximum Needs	Current Needs
P ₀	10	5
P ₁	4	2
P ₂	9	2

Before T₀:
3 drives available

Safe sequence
<P₁, P₀, P₂>

- At time **T₀** the system is in a safe state
- P₁ can be given 2 tape drives
- When P₁ releases its resources; there are 5 drives
- P₀ uses 5 and subsequently releases them (# 10 now)
- P₂ can then proceed

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.30

Example: 12 Tape drives available in the system

	Maximum Needs	Current Needs
P ₀	10	5
P ₁	4	2
P ₂	9	2

Before T1:
3 drives available

- At time **T1**, P₂ is allocated 1 tape drive

Example: 12 Tape drives available in the system

	Maximum Needs	Current Needs
P ₀	10	5
P ₁	4	2
P ₂	9	3

After T1:
2 drives available

- At time **T1**, P₂ is allocated 1 tape drive
- Only P₁ can proceed.
- When P₁ releases its resources; there are 4 drives
 - P₀ needs 5 and P₂ needs 6
- Mistake** in granting P₂ additional tape drive

Crux of deadlock avoidance algorithms

- **Ensure** that the system will always remain in a safe state
- Resource allocation request **granted** only if it will leave the system in a safe state

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.33

RESOURCE ALLOCATION GRAPH ALGORITHM

October 16, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.34

Claim edges

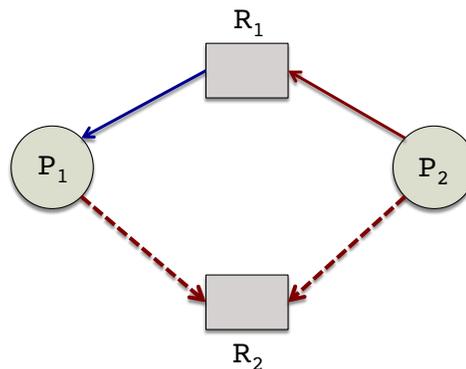
- Indicates that a process P_i may request a resource R_j at some time in the future.
- Representation:
 - ▣ Same direction as request
 - ▣ Dotted line

October 16, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.35

Resource allocation graph with a claim edge



October 16, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.36

Conversion of claim edges

- When process P_i requests resource R_j
 - Claim edge converted to a request edge
- When resource R_j released by P_i
 - The assignment edge $R_j \rightarrow P_i$ is **reconverted** to a claim edge $P_i \rightarrow R_j$

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.37

Allocating resources

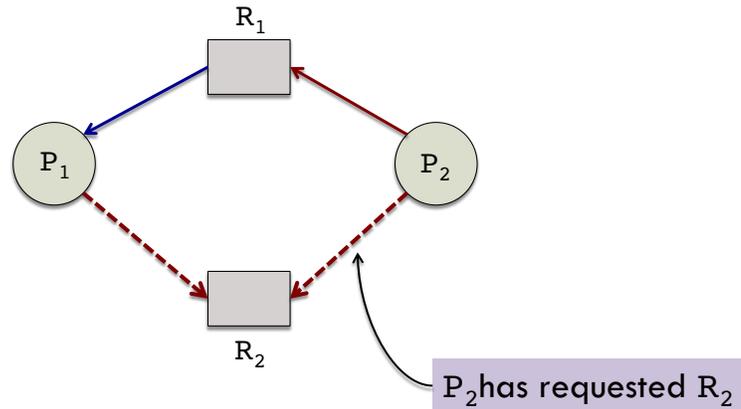
- When process P_i requests resource R_j
- Request granted only if
 - Converting claim edge to $P_i \rightarrow R_j$ to an assignment edge $R_j \rightarrow P_i$ **does not result** in a **cycle**

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.38

Using the allocation graph to allocate resources safely

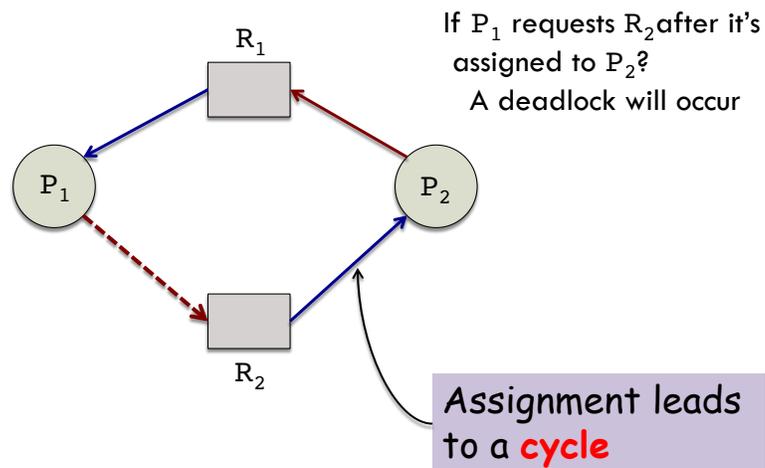


October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.39

Using the allocation graph to allocate resources safely



If P_1 requests R_2 after it's assigned to P_2 ?
A deadlock will occur

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.40

Resource allocation graph algorithm

- Not applicable in systems with multiple resource instances

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.41

The contents of this slide-set are based on the following references

- *Avi Silberschatz, Peter Galvin, Greg Gagne. Operating Systems Concepts, 9th edition. John Wiley & Sons, Inc. ISBN-13: 978-1118063330. [Chapter 7]*
- *Andrew S Tanenbaum. Modern Operating Systems. 4th Edition, 2014. Prentice Hall. ISBN: 013359162X/ 978-0133591620. [Chapter 6]*

October 16, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L17.42