# Operatines ignifest to the same Help

https://powcoder.com

Lecture 6b

# Previously

#### Deadlocks

- O Deadlock conditions Assignment Project Exam Help
- O Methods of handling deadlocks https://powcoder.com

## Today

#### More about deadlocks

- O More deadlock avoidangenment Project Exam Help
- O Resource Allocation Graphs
- O Deadlock detection and recovery powcoder.com

## Recap: Prevention of deadlocks

### Prevention by design

O Make sure that at least spenfield Project Exam Help conditions for a deadlock does not occur:

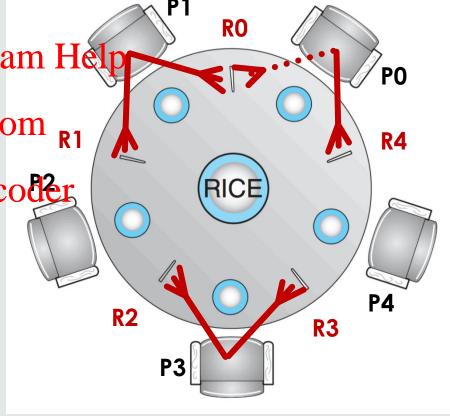
Mutual exclusion

O Hold-and-wait

O No-preemption

Circular wait

https://powcoder.com R1



### Recap: Avoidance of deadlocks

#### Resource Allocation Denial

Request[i] for process P<sub>i</sub>

Assignment Project Exam Help

- 1. If Request[i] <= Need[i], go to step 2.

  Otherwise, error (maximunhaxse/powcoder.com
- 2. If Request[i] < Available, go to step 3.
  Otherwise, P<sub>i</sub> must wait (resolute to the top of the policy of the po
- 3. Pretend to fulfill request:

```
Available := Available - Request[i]
Allocation[i] := Allocation[i] + Request[i]
Need[i] := Need[i] - Request[i]
```

Approve request if resulting state is safe. Otherwise, restore state and set P<sub>i</sub> to wait.

**Available** 

 $R_2$ 

# Recap: Avoidance of deadlocks

	Alloc	ation			Need				
	$R_0$	$R_1$	R <sub>2</sub>			$R_0$	$R_1$	$R_2$	
$P_0$	0	1	0		$P_0$	7	4	3	
P <sub>1</sub>	2	0	0 🛕	ssionn	ne <sup>P</sup> nt	Proje	ec <sup>2</sup> F	x 3m	Hel
$P_2$	3	0	2	ssignn	P <sub>2</sub>	6	0	0	1101
$P_3$	2	1	1	http	s: <sup>P</sup> / <sub>2</sub> n	owc	oder.	cdm	
$P_4$	0	0	2	Песр	P <sub>4</sub>	4	3	1	

O Assume request (1 0 2) bydd WeChat powcoder

# Recap: Avoidance of deadlocks

	Alloc	ation		
	$R_0$	$R_1$	$R_2$	
$P_0$	0	1	0	
P <sub>1</sub>	2	0	0 🛕	ssignn
$P_2$	3	0	2	
P <sub>3</sub>	2	1	1	http
$P_4$	0	0	2	P

		$R_0 R_1 R_2$			
	$P_0$	7	4	3	
ignn	nent i	Proje	ect E	xam	Help
	P <sub>2</sub>	6	0	0	•
http	s://p	owc	oder.	cdm	
•	$P_4$	4	3	1	

Available					
$R_0$	$R_1$	$R_2$			
3	3	2			

O Assume request (1 0 2) by dd We Chat powcodered

Allocation							
$R_0 R_1 R_2$							
$P_0$	0	1	0				
P <sub>1</sub>	3	0	2				
$P_2$	3	0	2				
$P_3$	2	1	1				
$P_4$	0	0	2				

Need						
$R_0 R_1 R_2$						
$P_0$	7	4	3			
P <sub>1</sub>	0	2	0			
$P_2$	6	0	0			
<b>P</b> <sub>3</sub>	0	1	1			
$P_4$	4	3	1			

Available					
$R_0$	R <sub>1</sub>	$R_2$			
2	3	0			

Sufficient resources to continue with P<sub>1</sub>

# Recap: Avoidance of deadlocks

Allocation						
$R_0 R_1 R_2$						
$P_0$	0	1	0			
P <sub>1</sub>	3	0	<sup>2</sup> A			
$P_2$	3	0	2			
$P_3$	2	1	1			
$P_4$	0	0	2			

		Need			
		$R_0 R_1 R_2$			
	$P_0$	7	4	3	
ignn	nent	Proje	ecť E	xam	H
http	s:///p	owc	oder.	cdm	
	P <sub>4</sub>	4	3	1	

Available					
$R_0 R_1 R_2$					
2	3	0			

- O Assume request (3 3 0) by P<sub>4</sub>
  Request cannot be granted (insufficient resources)
- O Assume request (0 2 0) by  $P_0$ Request cannot be granted (would result in unsafe state with only {2,1,0} available resources)

### Avoidance of deadlocks

#### Resource Initiation Denial

O Admit process P<sub>n</sub> onlywhen its resource jequests complete cause a deadlock

```
\sum_{i=0}^{n-1} \frac{\text{https://powcoder.com}}{\text{Need[i][j] + Need[n][j] <= Available[j]}} 
Add WeChat powcoder
```

- O Assumes that processes know in advance the amount of resources they will need
- O Assumes worst-case resource allocation (all processes request their maximum resources at the same time)

## Resource allocation graphs

### Graphical models of resource allocation

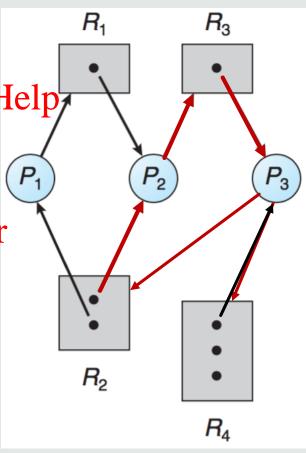
- O Process nodes P<sub>i</sub>
  Assignment Project Exam Help
- O Resource nodes R<sub>i</sub>
- O Number of instances of editips: pewspder.com

Add WeChat powcoder

- O Request edge P<sub>i</sub> → R<sub>j</sub>
- O Assignment edge R<sub>j</sub> → P<sub>i</sub>

What happens if  $P_3 \rightarrow R_2$ ?

Cycles in the graph indicate a deadlock.



# Resource allocation graphs

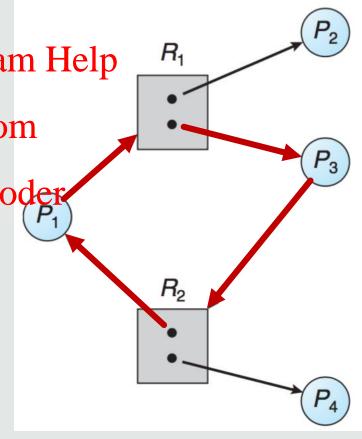
### Graphical models of resource allocation

- O Process nodes P<sub>i</sub>
  Assignment Project Exam Help
- O Resource nodes R<sub>i</sub>
- O Number of instances of edettps://pewspder.com

Add WeChat powcoder

- O Request edge P<sub>i</sub> → R<sub>j</sub>
- O Assignment edge R<sub>j</sub> → P<sub>i</sub>

Is there a deadlock in this system? Yes.



### **Deadlock Detection**

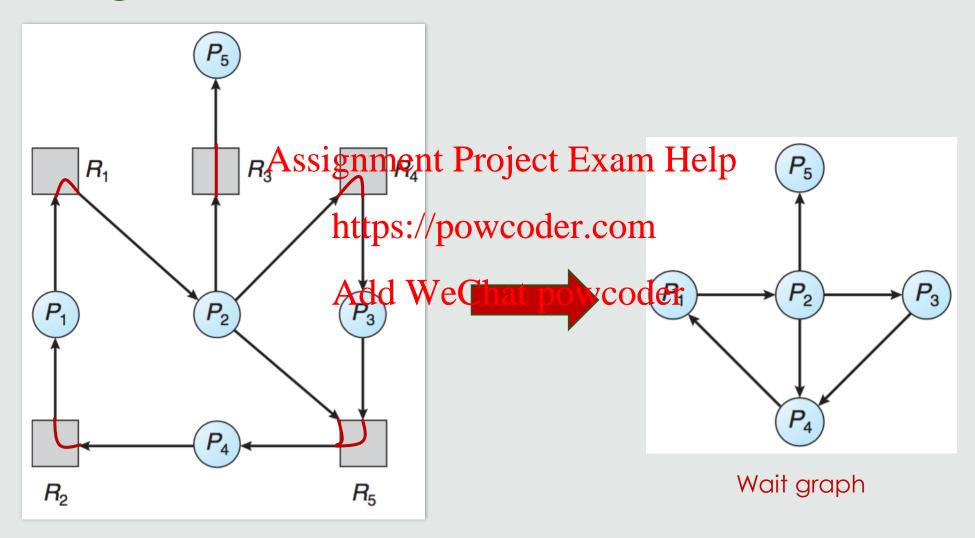
- O Always grant resource requests when resources are available
- O Examine the state of the system to determine whether a deadlock has occurred Assignment Project Exam Help
- O If so, recover from the deadlock

https://powcoder.com

O Drawbacks:

- O Runtime overhead for deadlock detection
- O Potential losses due to recovery

## Deadlock Detection – Single resources



Resource allocation graph

- O There are Available[j] instances of each resource type R<sub>j</sub>
- O Allocation[i][j] is the number of instances of resource type  $R_j$  held by process  $P_i$
- O Request[i][j] is the number of additional instances of resource type R<sub>j</sub> requested by process P<sub>i</sub>
- 2. Find an index i such that Add WeChat powcoder Finish[i] = false ^ Request[i] <= Work If no such i exists, go to step 4.
- 3. Work := Work + Allocation[i]
  Finish[i] := true
  Go to step 2.
- 4. If Finish[i] == false for some i, then the system is in a deadlocked state.

  P<sub>i</sub> is deadlocked if Finish[i] == false.

Allocation							
	$R_0 R_1 R_2$						
$P_0$	0	1	0				
P <sub>1</sub>	2	0	0				
$P_2$	3	0	3				
$P_3$	2	1	1				
$P_4$	0	0	2				

٨	Request Project				$\mathbf{D}_{2}$	Fin	ish
A	ssigi		R <sub>1</sub>			xam	пегр
	P <sub>0</sub> h1	ttns./	//p0v	vcod	er (	com P <sub>1</sub>	false
	P <sub>1</sub>	2	<b>P</b> 0	2		P <sub>1</sub>	false
	P <sub>2</sub> A	dd V	VeCl	naf p	ΟW	code	false
	P <sub>3</sub>	1	0	0	_ ,,	$P_3$	false
	$P_4$	0	0	2		$P_4$	false

Work					
$R_0$ $R_1$ $R_2$					
0	0	0			

Allocation								
$R_0 R_1 R_2$								
$P_0$	0	1	0					
P <sub>1</sub>	2	0	0					
$P_2$	3	0	3					
$P_3$	2	1	1					
$P_4$	0	0	2					

٨	Assignment Project				E,	Fin	ish
A	ssigi	R <sub>0</sub>	$R_1$			Xam	пер
	P <sub>0</sub> h1	ttng·	//nov	zchd	er (	വ്യ	false
	P <sub>1</sub>	2	/P0	2		com P <sub>1</sub>	false
	P <sub>2</sub> A	dd V	VeC1	nal p	ΟW	code	false
	$P_3$	1	0	0	. , ,	P <sub>3</sub>	false
	$P_4$	0	0	2		$P_4$	false

Work				
$R_0$ $R_1$ $R_2$				
0	0	0		

Allocation							
$R_0 R_1 R_2$							
$P_0$	0	1	0				
P <sub>1</sub>	2	0	0				
$P_2$	3	0	3				
$P_3$	2	1	1				
$P_4$	0	0	2				

Δ	Assignment Project				F	Fin	ish Lela
$\boldsymbol{\Lambda}$	sorg	R <sub>0</sub>	$R_1$	R <sub>2</sub>	رنال ، ا	Aam	ricip
	P <sub>0</sub> h1	ttns./	//nov	vc0d	er (	cden	true
	P <sub>1</sub>	2	/P0 V	2		com P <sub>1</sub>	false
	P <sub>2</sub>	dd V	VeC1	nal p	ΟW	code	false
	$P_3$	1	0	0	. , ,	P <sub>3</sub>	false
	$P_4$	0	0	2		P <sub>4</sub>	false

Work					
$R_0$ $R_1$ $R_2$					
0	1	0			

Allocation								
$R_0 R_1 R_2$								
$P_0$	0	1	0					
P <sub>1</sub>	2	0	0					
$P_2$	3	0	3					
$P_3$	2	1	1					
$P_4$	0	0	2					

٨	Assignment Project					Fin	ish
A	ssig	R <sub>0</sub>	$R_1$			xam	пец
	P <sub>0</sub> h1	ttns·/	//pov	vcod	er (	com P <sub>1</sub>	true
	P <sub>1</sub>	2	<b>P</b> 0	2		P <sub>1</sub>	false
	P <sub>2</sub> A	dd V	VeCl	nat p	ΟW	code	false
	$P_3$	1	0	0	_ ,,	$P_3$	false
	$P_4$	0	0	2		P <sub>4</sub>	false

Work					
$R_0 R_1 R_2$					
0	1	0			

Allocation								
$R_0 R_1 R_2$								
$P_0$	0	1	0					
P <sub>1</sub>	2	0	0					
$P_2$	0	0	0					
$P_3$	2	1	1					
$P_4$	0	0	2					

٨	Assignment Project					Fin	ish
A	ssig		R <sub>1</sub>			xam	пегр
	P <sub>0</sub> h1	ttns·	//p0v	vcod	er (	ငက်က	true
	P <sub>1</sub>	2	/P0	2		com P <sub>1</sub>	false
	P <sub>2</sub>	dd V	VeC1	naf p	ΟW	code	true
	$P_3$	1	0	0	. , ,	P <sub>3</sub>	false
	$P_4$	0	0	2		$P_4$	false

Work				
$R_0$ $R_1$ $R_2$				
3	1	3		

### Example 1

Allocation								
	$R_0 R_1 R_2$							
$P_0$	0	0	0					
P <sub>1</sub>	0	0	0					
$P_2$	0	0	0					
$P_3$	0	0	0					
$P_4$	0	0	0					

٨	Assignment Project				Fin	ish	
A	ssig		R <sub>1</sub>			XaIII	пегр
	P <sub>0</sub> h1	ttps:/	//pov	vcod	er.	com P <sub>1</sub>	true
	P <sub>1</sub>	0	<b>P</b> <sub>0</sub> •	0		P <sub>1</sub>	true
	P <sub>2</sub> A	dd V	VeCl	nat p	OW	code	true
	$P_3$	0	0	0		$P_3$	true
	$P_4$	0	0	0		$P_4$	true

Work				
$R_0 R_1 R_2$				
7	2	6		

All requests served – no deadlock

Allocation							
$R_0 R_1 R_2$							
$P_0$	0	1	0				
P <sub>1</sub>	2	0	0				
$P_2$	3	0	3				
$P_3$	2	1	1				
P <sub>4</sub>	0	0	2				

Δ	Assignment Project					Fin	ish Lela
$\boldsymbol{\Lambda}$	sorg	R <sub>0</sub>	$R_1$	R <sub>2</sub>	د نال	Aam	ricip
	P <sub>0</sub> h1	ttns./	//nov	vcod	er (	com	false
	P <sub>1</sub>	2	/P0 V	2		P <sub>1</sub>	false
	P <sub>2</sub>	dd V	VeC1	nat p	ΟW	code	false
	$P_3$	1	0	0	• , ,	P <sub>3</sub>	false
	$P_4$	0	0	2		P <sub>4</sub>	false

Work				
$R_0 R_1 R_2$				
0	0	0		

Allocation								
	$R_0 R_1 R_2$							
$P_0$	0	1	0					
P <sub>1</sub>	2	0	0					
$P_2$	3	0	3					
$P_3$	2	1	1					
$P_4$	0	0	2					

Δ	Assignment Project					Fin	ish
$\boldsymbol{\Lambda}$	soig	R <sub>0</sub>	$R_1$	R <sub>2</sub>	د نسال	Xaiii .	ricip
	P <sub>0</sub> h1	ttns·/	//nov	vcod	er (	com	false
	P <sub>1</sub>	2	/P0 V	2		P <sub>1</sub>	false
	P <sub>2</sub>	dd V	VeC1	nat p	ΟW	code	false
	$P_3$	1	0	0	• , ,	P <sub>3</sub>	false
	$P_4$	0	0	2		P <sub>4</sub>	false

Work					
$R_0 R_1 R_2$					
0	0	0			

Allocation								
	$R_0 R_1 R_2$							
$P_0$	0	1	0					
P <sub>1</sub>	2	0	0					
$P_2$	3	0	3					
$P_3$	2	1	1					
P <sub>4</sub>	0	0	2					

٨	Assignment Project				E	Fin	ish
A	ssig	R <sub>0</sub>	$R_1$			XaIII	rieip
	P <sub>0</sub> h1	ttns	//pov	vc6d	er (	cden	true
	P <sub>1</sub>	2	/P0	2		com P <sub>1</sub>	false
	P <sub>2</sub>	dd V	VeCl	naf p	ΟW	code	false
	$P_3$	1	0	0	_ ,,	P <sub>3</sub>	false
	$P_4$	0	0	2		P <sub>4</sub>	false

Work			
$R_0$	$R_1$	R <sub>2</sub>	
0	1	0	

#### Example 2

Allocation				
	$R_0$	$R_1$	$R_2$	
$P_0$	0	1	0	
P <sub>1</sub>	2	0	0	
$P_2$	3	0	3	
$P_3$	2	1	1	
$P_4$	0	0	2	

Δ	ssio	Request Project			Finish		ish Helr
<i>1</i> 3	133151	R <sub>0</sub>	$R_1$	R <sub>2</sub>		XuIII	ricip
	P <sub>0</sub> h1	ttns·/	$\frac{1}{2}$	vcod	er (	com	true
	P <sub>1</sub>	2	<b>P</b> 0	2		P <sub>1</sub>	false
	P <sub>2</sub> A	dd V	VeCl	nat p	ΟW	code	false
	$P_3$	1	0	0		$P_3$	false
	P <sub>4</sub>	0	0	2		P <sub>4</sub>	false

Work			
$R_0$	$R_1$	$R_2$	
0	1	0	

No further requests can be served - Processes P<sub>1</sub>, ..., P<sub>4</sub> deadlocked

### **Deadlock Detection**

When to do deadlock detection?

- O How often is a deadle challengment enterprise Exam Help
- O How many processes will be affected by deadlock when it happens? https://powcoder.com
- O Run it

- O on every resource request?
- O every hour?
- O based on metrics?

# Deadlock Recovery

- O Abort
  - O all processes involved in deadlock
  - o individual processes Astilithere is no Project Exam Help
- O Rollback
  - o all processes involved in dettps://powcoder.com
  - o individual processes until there is we deadlock anymere.
- O How to choose which processes to kill or preempt?
  - O Minimise "loss"
  - O Least CPU time?
  - O Fewest allocated resources?
- O Deadlock could happen again

# Combined Deadlock Handling Strategy

#### For example:

- O Group resources into resource types:
  - O Disks Assignment Project Exam Help
  - O I/O devices
  - O Files
  - O Main memory

Add WeChat powcoder

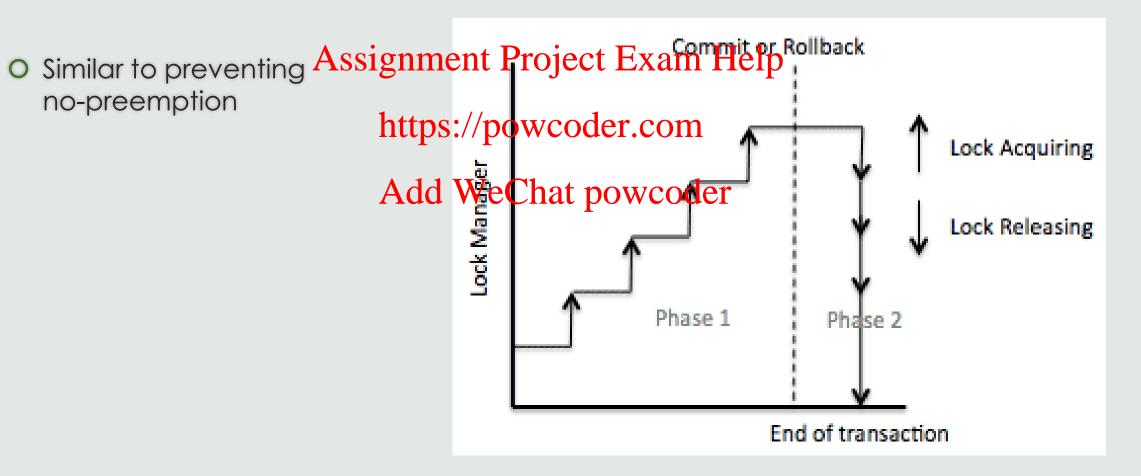
https://powcoder.com

- O Prevent circular wait between these classes
- O Use appropriate methods within these classes, e.g.
  - O Prevent no-preemption on main memory
  - O Avoidance on I/O devices
  - 0 . . .

# Two-Phase Locking

As used in databases

no-preemption



# Summary

#### Deadlocks

- O Methods for handling Adexig trackent Project Exam Help
  - O Deadlock prevention
- https://powcoder.com
- O Deadlock avoidance
- O Deadlock detection and MedweChat powcoder

### Read

- O Tanenbaum & Bos., Modern Operating Systems
  - O Chapter 6

Assignment Project Exam Help

- O Silberschatz et al., Operatihttps://epowooden.com
  - O Chapter 7

### **Next Lecture**

- O Introduction O Deadlocks
- O Operating System Architectures Assignment Project Exam Help
- O Processes O File Systems
- o Threads Programming https://powcederscombutput
- O Process Scheduling Evaluation WeCharpsecurity and Virtualisation
- O Process Synchronisation