Review

- Deadlock detection and Recovery.
  - Detection with one resource of each type
  - Detection with multiple resource of each type
    - Detection Algorithm with multiple matrix
      - Maintain four matrices
        - E (Existing), A(Available), C (Currently holding), R (requiest)
  - Recover from Deadlock after a deadlock detection
    - Recovery by preemption
    - Recovery by rollback
    - Recovery through killing processes.

Preview

- Deadlock Avoidance
  - Safe and Unsafe State
  - Banker's Algorithm for a Single Resource per each Type
  - Banker's Algorithm for Multiple Resource per each Type
- Deadlock Prevention
  - Attack Mutual Exclusion
  - Attack Hold and Wait
  - Attack No Preemption
  - Attack Circular Wait

Deadlock Avoidance

- When a process request a resource, OS must be able to decide whether granting a resource is safe or not and only make allocation when it is safe.
- Is there any algorithm that can always avoid deadlock by making the right choice all the time?
  - Yes, with certain information is available in advance

Deadlock Avoidance
(Safe and Unsafe State)

A state is said safe state

- If it is not deadlocked and
- There is some scheduling order in which every process can run to completion even <u>if all of them suddenly request their maximum number of resources immediately</u>.

Banker's Algorithm for a Single Resource (Dijkstra)

- Modeled on the way a small-town banker might need to deal with a group of customer.
- Banker provide a credit for each customer based on his/her credit score.
- Bank does not need prepared the total sum of all customer credit.
- What banker's algorithm does is <u>check to see whether granting the request leads to a unsafe state or not.</u>
- <u>If granting the request leads to an unsafe state</u>, the request is denied.

Banker's Algorithm for a Single Resource (Dijkstra)

Ex)

- There are four customers A, B, C, D, each of whom has been granted a certain amount of credit units.

	HAS	MAX
A	0	6
B	0	5
C	0	4
D	0	7

a) Free 10

	HAS	MAX
A	1	6
B	1	5
C	2	4
D	4	7

b) Free 2

	HAS	MAX
A	1	6
B	2	5
C	2	4
D	4	7

c) Free 1

- The banker knows that not all customers will need their maximum credit immediately, but banker has reserve only 10 units rather than 22.
- At a certain moment, the situation is as shown b) or c)

Banker's Algorithm for a Single Resource (Dijkstra)

	HAS	MAX
A	0	6
B	0	5
C	0	4
D	0	7

Free 10

	HAS	MAX
A	1	6
B	1	5
C	2	4
D	4	7

Free 2

B request one more resource

UNSAFE

	HAS	MAX
A	1	6
B	2	5
C	2	4
D	4	7

Free 1

Deadlock Avoidance
(Safe and Unsafe State)

Process B request 2 resource

	HAS	MAX
A	3	9
B	2	4
C	2	7

FREE 3

	HAS	MAX
A	3	9
B	4	4
C	2	7

FREE 1

	HAS	MAX
A	3	9
B	0	-
C	2	7

FREE 5

	HAS	MAX
A	3	9
B	0	-
C	7	7

FREE 0

	HAS	MAX
A	3	9
B	0	-
C	0	-

FREE 7

safe

Deadlock Avoidance
(Safe and Unsafe State)

Process A request 1 resource

	HAS	MAX
A	3	9
B	2	4
C	2	7

FREE 3

	HAS	MAX
A	4	9
B	2	4
C	2	7

FREE 2

	HAS	MAX
A	4	9
B	4	4
C	2	7

FREE 0

	HAS	MAX
A	4	9
B	0	-
C	2	7

FREE 4

Unsafe

Banker's Algorithm for Multiple Resource

Using three matrix for checking a safe state

- Available resource matrix **(A)** – Present how may number of resources per each type are available at any moment, since some of resources are assigned to processes are not available
- Request matrix (R) – present how many resources are needed for processes to finish their job.
- Current allocation matrix (C) – present how resources are currently held by processes

Banker's Algorithm for Multiple Resource

The algorithm for checking to see if a state is safe or not

1. Look for a row of vector R, whose unmet resource needs are all smaller than or equal to A. If no such row exists, the system will eventually deadlock since no process can run to completion.
2. Assume the process of the row chosen requests all the resources it needs and finishes. Mark that process as terminated and add all its resources to the A vector.
3. repeat step 1 and 2 until either all processes are marked terminated (safe state) or until a deadlock occurs (unsafe).

Banker's Algorithm for Multiple Resource

Process	Tape drives	Plotters	Scanners	CD ROMs
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Scanners	CD ROMs
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Resources still needed

E = (6342)
P = (5322)
A = (1020)

Banker's Algorithm for Multiple Resource

- E = (6, 3, 4, 2)
- A = (1, 0, 2, 0)

$$C = \begin{bmatrix} 3011 \\ 0100 \\ 1110 \\ 1101 \\ 0000 \end{bmatrix}, R = \begin{bmatrix} 1100 \\ 0112 \\ 3100 \\ 0010 \\ 2110 \end{bmatrix}$$

Possible requests
- P_1 request one R_1
- P_2 request one R_3
- P_3 request one R_1
- P_4 request one R_3
- P_5 request one R_1
- P_5 request one R_3

Banker's Algorithm for Multiple Resource

- E = (6, 3, 4, 2)
- A = (1, 0, 2, 0)

$$C = \begin{bmatrix} 3011 \\ 0100 \\ 1110 \\ 1101 \\ 0000 \end{bmatrix}, R = \begin{bmatrix} 1100 \\ 0112 \\ 3100 \\ 0010 \\ 2110 \end{bmatrix}$$

Possible requests
- P_1 request one R_1

Banker's Algorithm for Multiple Resource

- E = (6, 3, 4, 2)
- A = (0, 0, 2, 0)

$$C = \begin{bmatrix} 4011 \\ 0100 \\ 1110 \\ 1101 \\ 0000 \end{bmatrix}, R = \begin{bmatrix} 0100 \\ 0112 \\ 3100 \\ 0010 \\ 2110 \end{bmatrix}$$

Possible requests
- P_1 request one R_1

$$\begin{array}{l} A = (0, 0, 2, 0) \\ \quad\downarrow P_4 \\ A = (0, 0, 2, 0)+(1,1,0,1) = (1,1,2,1) \\ \quad\downarrow P_1 \\ A = (1, 1, 2, 1)+(4,0,1,1) = (5,1,3,2) \\ \quad\downarrow P_2 \\ A = (5, 1, 3, 2)+(0,1,0,0) = (5,2,3,2) \\ \quad\downarrow P_3 \\ A = (5, 2, 3, 2)+(1,1,1,0) = (6,3,4,2) \\ \quad\downarrow P_5 \\ A = (6, 3, 4, 2)+(0,0,0,0) = (6,3,4,2) \end{array}$$

Banker's Algorithm for Multiple Resource

- E = (6, 3, 4, 2)
- A = (1, 0, 2, 0)

$$C = \begin{bmatrix} 3011 \\ 0100 \\ 1110 \\ 1101 \\ 0000 \end{bmatrix}, R = \begin{bmatrix} 1100 \\ 0112 \\ 3100 \\ 0010 \\ 2110 \end{bmatrix}$$

Possible requests
- P_5 request one R_1 one R_3

Banker's Algorithm for Multiple Resource

- E = (6, 3, 4, 2)
- A = (0, 0, 1, 0)

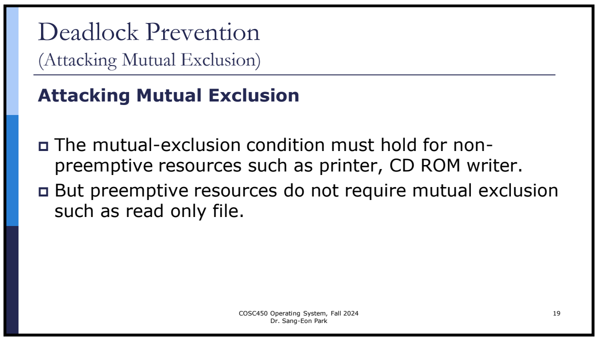
$$C = \begin{bmatrix} 3011 \\ 0100 \\ 1110 \\ 1101 \\ 1010 \end{bmatrix}, R = \begin{bmatrix} 1100 \\ 0112 \\ 3100 \\ 0010 \\ 1100 \end{bmatrix}$$

Possible requests
- P_5 request one R_1 one R_3

$$\begin{array}{l} A = (0, 0, 1, 0) \\ \quad\downarrow P_4 \\ A = (0, 0, 1, 0)+(1,1,0,1) = (1,1,1,1) \\ \quad\downarrow P_1 \\ A = (1, 1, 1, 1)+(3,0,1,1) = (4,1,2,2) \\ \quad\downarrow P_2 \\ A = (4, 1, 2, 2)+(0,1,0,0) = (4,2,2,2) \\ \quad\downarrow P_3 \\ A = (4, 2, 2, 2)+(1,1,1,0) = (5,3,3,2) \\ \quad\downarrow P_5 \\ A = (5, 3, 3, 2)+(1,0,1,0) = (6,3,4,2) \end{array}$$

Deadlock Prevention

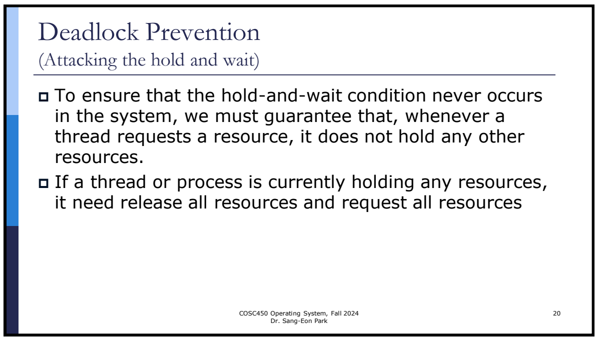
- Actually, deadlock avoidance is essentially impossible to use for OS in complex system, since it requires information about future requests which is usually not known.
- Deadlock prevention is a method to attack one of four deadlock condition.
  1. Mutual exclusion
  2. Hold and wait
  3. No preemption
  4. Circular wait



Deadlock Prevention
(Attacking Mutual Exclusion)

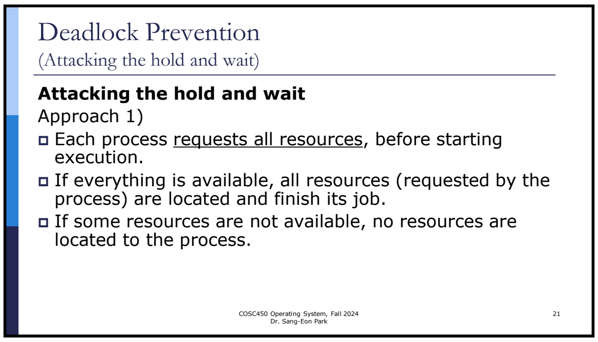
Attacking Mutual Exclusion

- The mutual-exclusion condition must hold for non-preemptive resources such as printer, CD ROM writer.
- But preemptive resources do not require mutual exclusion such as read only file.



Deadlock Prevention
(Attacking the hold and wait)

- To ensure that the hold-and-wait condition never occurs in the system, we must guarantee that, whenever a thread requests a resource, it does not hold any other resources.
- If a thread or process is currently holding any resources, it need release all resources and request all resources

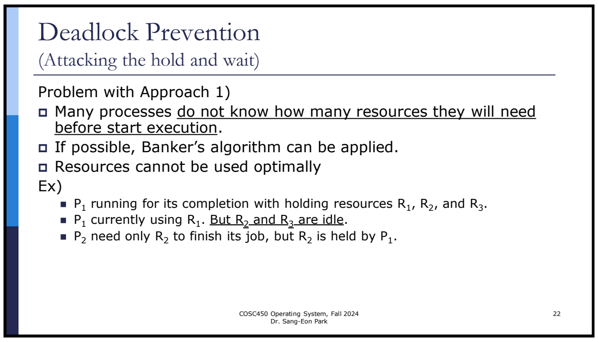


Deadlock Prevention
(Attacking the hold and wait)

Attacking the hold and wait

Approach 1)

- Each process <u>requests all resources</u>, before starting execution.
- If everything is available, all resources (requested by the process) are located and finish its job.
- If some resources are not available, no resources are located to the process.



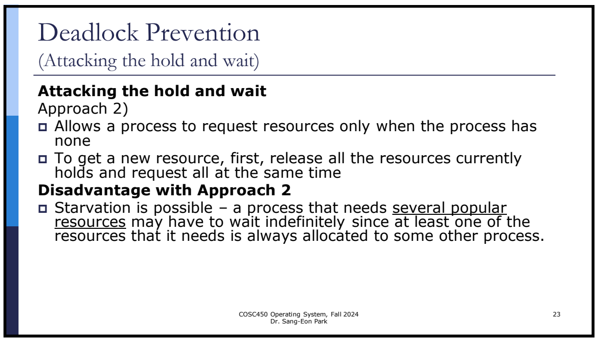
Deadlock Prevention
(Attacking the hold and wait)

Problem with Approach 1)

- Many processes <u>do not know how many resources they will need before start execution</u>.
- If possible, Banker's algorithm can be applied.
- Resources cannot be used optimally

Ex)

  - P_1 running for its completion with holding resources R_1, R_2, and R_3.
  - P_1 currently using R_1. <u>But R_2 and R_3 are idle</u>.
  - P_2 need only R_2 to finish its job, but R_2 is held by P_1.



Deadlock Prevention
(Attacking the hold and wait)

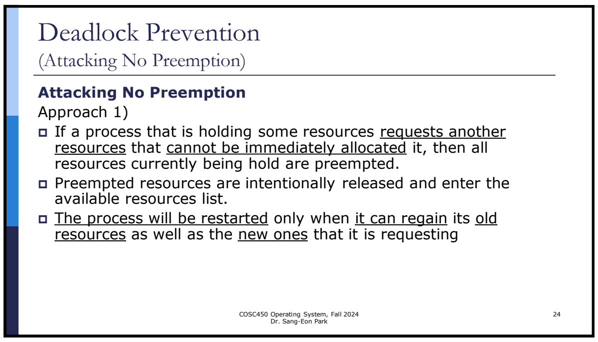
Attacking the hold and wait

Approach 2)

- Allows a process to request resources only when the process has none
- To get a new resource, first, release all the resources currently holds and request all at the same time

Disadvantage with Approach 2

- Starvation is possible – a process that needs <u>several popular resources</u> may have to wait indefinitely since at least one of the resources that it needs is always allocated to some other process.



Deadlock Prevention
(Attacking No Preemption)

Attacking No Preemption

Approach 1)

- If a process that is holding some resources <u>requests another resources</u> that <u>cannot be immediately allocated</u> it, then all resources currently being hold are preempted.
- Preempted resources are intentionally released and enter the available resources list.
- <u>The process will be restarted</u> only when <u>it can regain</u> its <u>old resources</u> as well as the <u>new ones</u> that it is requesting

Deadlock Prevention
(Attacking No Preemption)

Attacking No Preemption
Approach 2)
- If a process requests some resources, first check whether they are available or not
- If they are, allocated them to the process.
- If they are not available, check whether they are allocated to some other process that is waiting for additional resources.
- If so, preempt the desired resources from the waiting process and allocate them to the requesting process.
- If the resources are not either available or held by waiting process, the requesting process must wait.

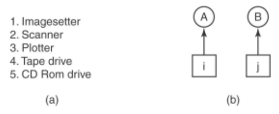
Deadlock Prevention
(Attacking the Circular Wait Condition)

Attacking the Circular Wait Condition

Approach 1)
- A process can hold only one resource.
- If a process need second resource, the process need release the first one.
- But sometimes this approach is not acceptable
  - ex) a process need a copy a huge file from the disk.
  - A process need hold buffer and disk for read and write.

Approach 2)
- Global number is provided to each resource
- A process can request resources whenever they want, but all requests must made in numerical order.

Deadlock Prevention
(Attacking the Circular Wait Condition)



- Only deadlock can occur the process A request resource j and the process B request the resource i. but it never can happen since either i > j or i < j