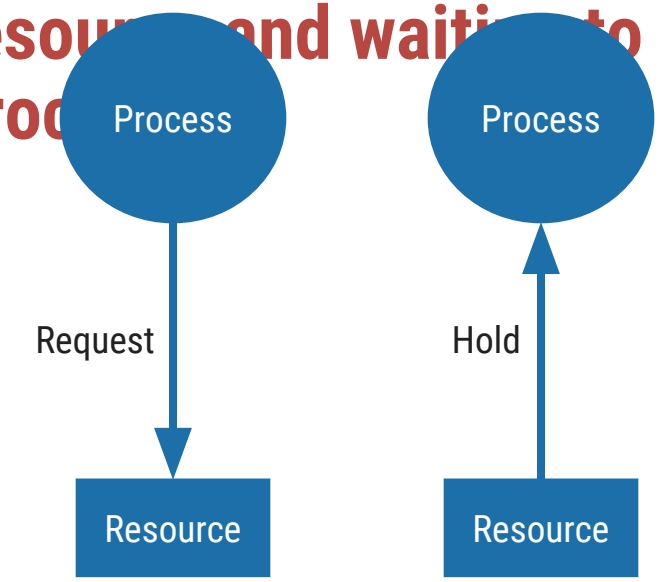
Deadlock

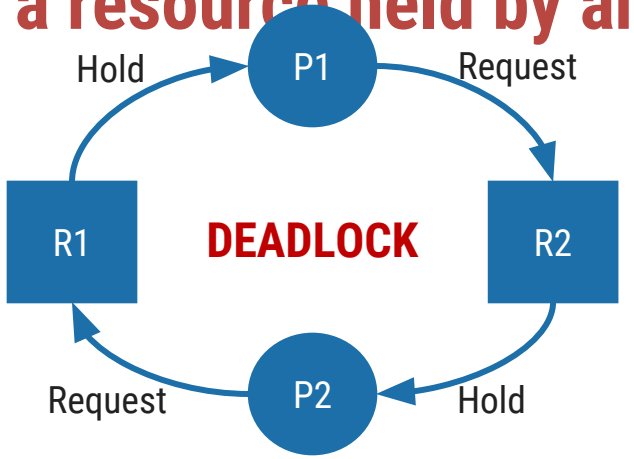
Outline

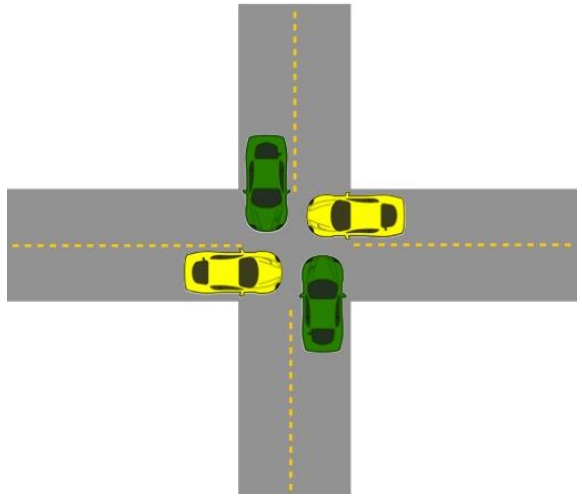
- Basic concepts of Deadlock
- Deadlock characteristics
- Deadlock ignorance
  - Ostrich algorithm
- Deadlock detection and recovery
- Deadlock avoidance
  - Banker’s algorithm
- Deadlock prevention

What is Deadlock?

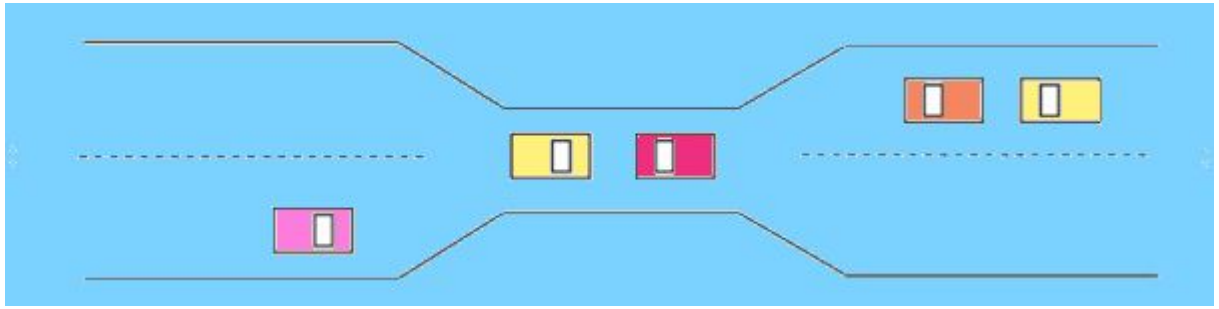
- A set of processes is deadlocked if **each process in the set is waiting for an event that only another process in the set can cause**.
- Deadlocks are a **set of blocked processes each holding a resou and waiti to acquire a resource held by another proc**







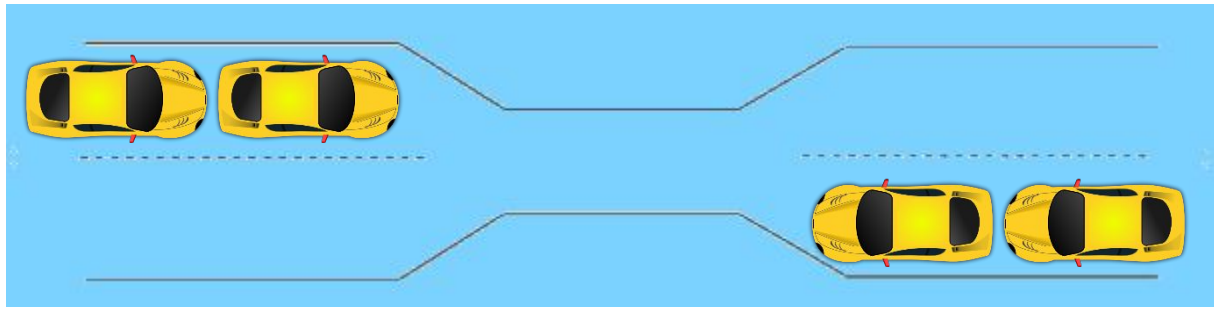
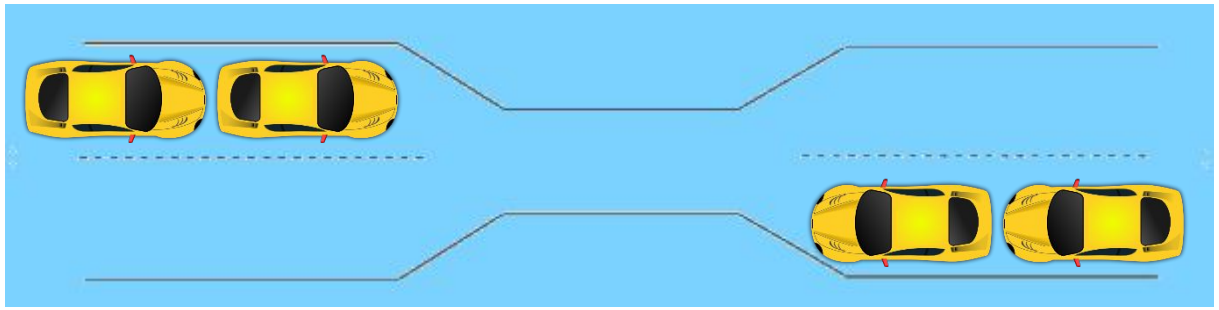
Exercise



Preemptable and non-preemptable resource

- Preemptable:- Preemptive resources are those **which can be taken away from a process without causing any ill effects** to the process.
  - Example:- Memory.
- Non-preemptable:- Non-pre-emptive resources are those **which cannot be taken away from the process without causing any ill effects** to the process.
  - Example:- CD-ROM (CD recorder), Printer.

Deadlock v/s Starvation

Deadlock	Starvation
	

Deadlock v/s Starvation

Deadlock	Starvation
All processes keep waiting for each other to complete and none get executed.	High priority process keep executing and low priority process are blocked.
Resources are blocked by the process.	Resources are continuously utilized by the higher priority process.
Necessary conditions are mutual exclusion, hold and wait, no preemption, circular wait.	Priorities are assigned to the process.
Also known as circular wait.	Also known as lived lock.
It can be prevented by avoiding the necessary conditions for deadlock.	It can be prevented by Aging.

Conditions that lead to deadlock (Deadlock characteristics)

1. Mutual exclusion
   - **Each resource is either** currently **assigned to exactly one process** or **is available**.
   - Only **one process at a time can use a resource.**
2. Hold and wait
   - Process currently holding resources granted earlier can **request more resources**.
3. No preemption
   - Previously granted resources **cannot be forcibly taken away** from process.
4. Circular wait
   - There must be a **circular chain of 2 or more processes**. Each process is waiting for resource that is held by next member of the chain.

- **All four of these conditions must be present for a deadlock to occur.**

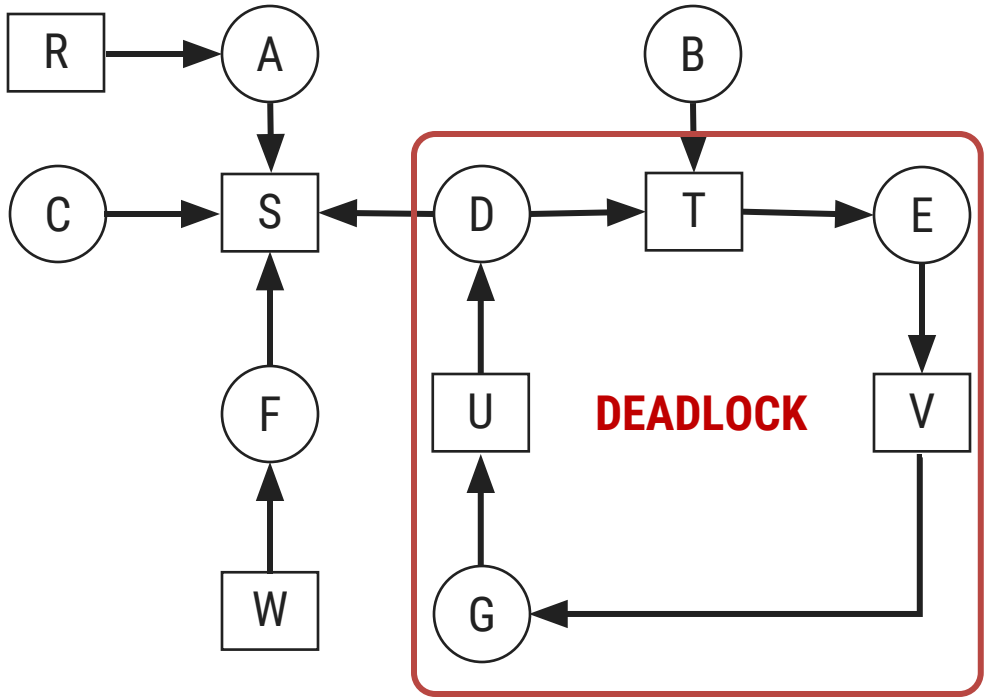
Strategies for dealing with deadlock

1. Just **ignore** the problem
2. **Detection and recovery**.
   - Let deadlocks occur, detect them and take action.
3. Dynamic **avoidance** by careful resource allocation.
4. **Prevention**, by structurally negating (killing) one of the four required conditions.

Deadlock ignorance (Ostrich Algorithm)

- When **storm approaches**, an **ostrich puts his head in the sand (ground)** and **pretend (imagine) that there is no problem at all**.
- **Ignore** the **deadlock** and **pretend** that **deadlock never occur**.
- Reasonable if
  - deadlocks occur very rarely
  - difficult to detect
  - cost of prevention is high
- **UNIX** and **Windows** takes this approach

Deadlock detection for single resource (RAG - Resource Allocation Graph)



- We are starting from node D.
- Empty list L = ()
- Add current node so Empty list = (D).
- From this node there is one outgoing arc to T so add T to list.
- So list become L = (D, T).
- Continue this step….so we get list as below
  L = (D, T, E)…… L = (**D**, T, E, V, G, U, **D**)
- In the above step in list the node **D appears twice**, **so deadlock**.

Deadlock detection for single resource (RAG - Resource Allocation Graph)

- Algorithm for detecting deadlock for single resource
  - For each node, N in the graph, perform the following five steps with N as the starting node.
    1) Initialize L to the empty list, designate all arcs as unmarked.
    2) Add current node to end of L, check to see if node now appears in L two times. If it does, graph contains a cycle (listed in L), algorithm terminates.
    3) From given node, see if any unmarked outgoing arcs. If so, go to step 4; if not, go to step 5.
    4) Pick an unmarked outgoing arc at random and mark it. Then follow it to the new current node and go to step 2.
    5) If this is initial node, graph does not contain any cycles, algorithm terminates. Otherwise, dead end. Remove it, go back to previous node, make that one current node, go to step 2.

Deadlock detection for multiple resources

T =

Tape Drives	Plotters	Scanners	CD Roms
4	2	3	1

total no of each resource

A =

Tape Drives	Plotters	Scanners	CD Roms
2	1	0	0

no of resources that are available (free)

C =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	1	0
P2	2	0	0	1
P3	0	1	2	0

no of resources held by each process

R =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	2	0	0	1
P2	1	0	1	1
P3	2	1	0	0

no of resources still needed by each process to proceed

Deadlock detection for multiple resources

T =

Tape Drives	Plotters	Scanners	CD Roms
4	2	3	1

total no of each resource

A =

Tape Drives	Plotters	Scanners	CD Roms
0	0	0	0

no of resources that are available (free)

C =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	1	0
P2	2	0	0	1
P3	2	2	2	0

no of resources held by each process

R =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	2	0	0	1
P2	1	0	1	1
P3	2	1	0	0

no of resources still needed by each process to proceed

Deadlock detection for multiple resources

T =

Tape Drives	Plotters	Scanners	CD Roms
4	2	3	1

total no of each resource

A =

Tape Drives	Plotters	Scanners	CD Roms
2	2	2	0

no of resources that are available (free)

C =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	1	0
P2	2	0	0	1
~~P3~~	~~0~~	~~0~~	~~0~~	~~0~~

no of resources held by each process

DEADLOCK

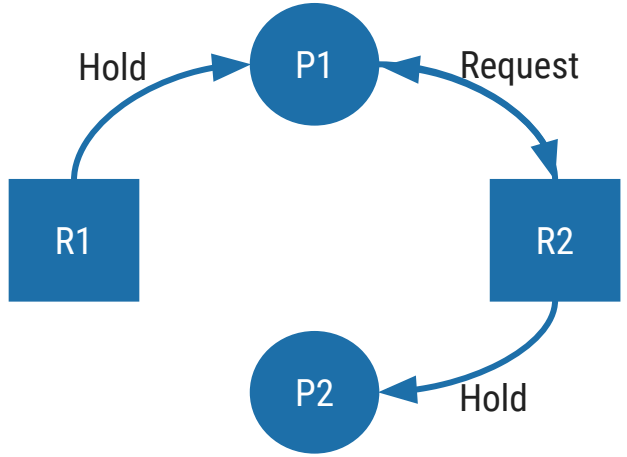
R =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	2	0	0	(1)
P2	1	0	(1)	(1)
~~P3~~	~~2~~	~~1~~	~~0~~	~~0~~

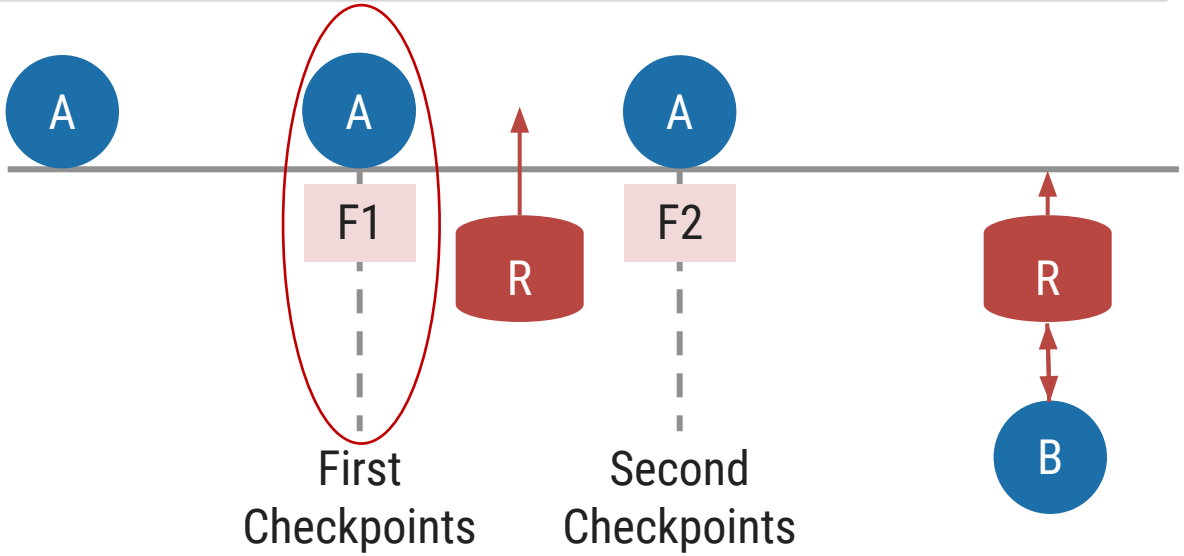
no of resources still needed by each process to proceed

Deadlock recovery

1. Recovery through pre-emption
   - In this method **resources are temporarily taken away from its current owner and give it to another process**.
   - The ability to take a resource away from a process, have another process use it, and then give it back without the process noticing it is **highly dependent on the nature of the resource**.
   - Recovering this way is frequently **difficult or impossible**.

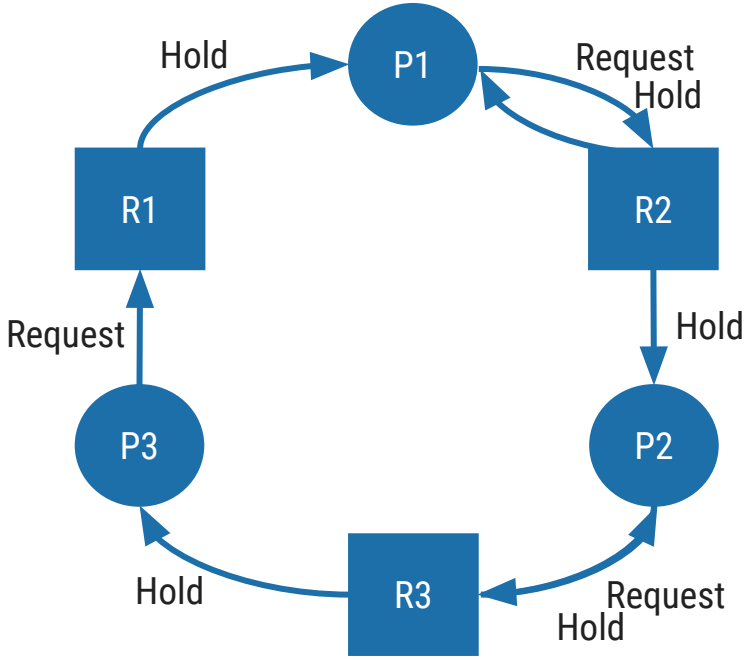


2. Recovery through rollback
   - **PCB (Process Control Block)** and **resource state** are **periodically saved** at "**checkpoint**".
   - When **deadlock is detected**, **rollback the preempted process up to the previous safe state** before it acquired that resource.
   - **Discard the resource** manipulation that occurred after that checkpoint.
   - **Start the process after it is determined** it can run again.



Deadlock recovery

3. Recovery through killing processes
   - The simplest way to break a deadlock is **to kill one or more processes**.
   - Kill all the process involved in deadlock
   - Kill process one by one.
   - After killing each process check for deadlock
     - If deadlock recovered then stop killing more process
     - Otherwise kill another process



Safe and unsafe states

- A state is said to be safe **if it is not deadlocked** and **there is some scheduling order in which every process can run to completion** even if all of them suddenly request their maximum number of resources immediately.
- Total resources are 10
- 7 resources already allocated
- So there are 3 still free
- A need 6 resources more to complete it.
- B need 2 resources more to complete it.
- C need 5 resources more to complete it.

Process	Has / Hold	Max required
A	3	9
B	2	4
C	2	7
Free : 3		

Safe state

Process	Has / Hold	Max required
A	3	9
B	2	4
C	2	7
Free : 3		

2

Process	Has / Hold	Max required
A	3	9
B	4	4
C	2	7
Free : 1		

4

Process	Has / Hold	Max required
A	3	9
B	0	-
C	2	7
Free : 5		

5

Process	Has / Hold	Max required
A	3	9
B	0	
C	7	7
Free : 0		

7

Process	Has / Hold	Max required
A	3	9
B	0	-
C	0	-
Free : 7		

6

Process	Has / Hold	Max required
A	9	9
B	0	-
C	0	-
Free : 1		

Unsafe state

Process	Has / Hold	Max required
A	3	9
B	2	4
C	2	7
Free : 3		

1

Process	Has / Hold	Max required
A	4	9
B	2	4
C	2	7
Free : 2		

2

Process	Has / Hold	Max required
A	4	9
B	4	4
C	2	7
Free : 0		

4

Process	Has / Hold	Max required
A	4	9
B	0	-
C	2	7
Free : 4		

Deadlock avoidance

- Deadlock can be avoided by **allocating resources carefully**.
- Carefully **analyze each resource request** to see **if it can be safely granted**.
- Need an algorithm that can always avoid deadlock by making right choice all the time **(Banker's algorithm).**
- Banker's algorithm for single resource
- Banker's algorithm for multiple resource

Banker's algorithm for single resource

- What the algorithm does is **check to see if granting the request leads to an unsafe state**. If it does, the **request is denied**.
- If **granting the request leads to a safe state**, it is **carried out**.
- If we have situation as per figure
  - then it is safe state
  - because with 10 free units
  - one by one all customers can be served.

Process	Has / Hold	Max required
A	0	6
B	0	5
C	0	4
D	0	7
Free : 10		

Banker's algorithm for single resource (safe state)

Process	Has / Hold	Max required
A	1	6
B	1	5
C	2	4
D	4	7
Free : 2		

2

Process	Has / Hold	Max required
A	1	6
B	1	5
C	4	4
D	4	7
Free : 0		

Process	Has / Hold	Max required
A	1	6
B	1	5
C	0	-
D	4	7
Free : 4		

3

Process	Has / Hold	Max required
A	1	6
B	1	5
C	0	-
D	7	7
Free : 1		

Process	Has / Hold	Max required
A	1	6
B	1	5
C	0	-
D	0	-
Free : 8		

4

Process	Has / Hold	Max required
A	1	6
B	5	5
C	0	-
D	0	-
Free : 4		

Banker’s algorithm for single resource (safe state)

Process	Has / Hold	Max required
A	1	6
B	0	-
C	0	-
D	0	-
Free : 9		

5

Process	Has / Hold	Max required
A	6	6
B	0	-
C	0	-
D	0	-
Free : 4		

Process	Has / Hold	Max required
A	0	-
B	0	-
C	0	-
D	0	-
Free : 10		

- The order of execution is C, D, B, A. So if we can find proper order of execution then there is no deadlock.

Banker’s algorithm for single resource (unsafe state)

Process	Has / Hold	Max required
A	1	6
B	2	5
C	2	4
D	4	7
Free : 1		

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
1	0	2	0

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	3	0	1	1
P2	0	1	0	0
P3	1	1	1	0
P4	1	1	0	1
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	1	1	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	1	0
P5	2	1	1	0

no of resources still needed by each process to proceed

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
1	0	1	0

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	3	0	1	1
P2	0	1	0	0
P3	1	1	1	0
P4	1	1	1	1
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	1	1	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	0	0
P5	2	1	1	0

no of resources still needed by each process to proceed

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
2	1	2	1

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	3	0	1	1
P2	0	1	0	0
P3	1	1	1	0
P4	-	-	-	-
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	1	1	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	0	0
P5	2	1	1	0

no of resources still needed by each process to proceed

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
1	0	2	1

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	4	1	1	1
P2	0	1	0	0
P3	1	1	1	0
P4	-	-	-	-
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	0	0
P5	2	1	1	0

no of resources still needed by each process to proceed

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
5	1	3	2

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	-	-	-	-
P2	0	1	0	0
P3	1	1	1	0
P4	-	-	-	-
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	0	0
P5	2	1	1	0

no of resources still needed by each process to proceed

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
5	0	2	0

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	-	-	-	-
P2	0	2	1	2
P3	1	1	1	0
P4	-	-	-	-
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	0	0
P2	0	0	0	0
P3	3	1	0	0
P4	0	0	0	0
P5	2	1	1	0

no of resources still needed by each process to proceed

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
5	2	3	2

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	-	-	-	-
P2	-	-	-	-
P3	1	1	1	0
P4	-	-	-	-
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	0	0
P2	0	0	0	0
P3	3	1	0	0
P4	0	0	0	0
P5	2	1	1	0

no of resources still needed by each process to proceed

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
2	1	3	2

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms	no of resources held by each process
P1	-	-	-	-	
P2	-	-	-	-	
P3	4	2	1	0	
P4	-	-	-	-	
P5	0	0	0	0	

Process	Tape Drives	Plotters	Scanners	CD Roms	no of resources still needed by each process to proceed
P1	0	0	0	0	
P2	0	0	0	0	
P3	0	0	0	0	
P4	0	0	0	0	
P5	2	1	1	0	

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	-	-	-	-
P2	-	-	-	-
P3	-	-	-	-
P4	-	-	-	-
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	0	0	0	0
P5	2	1	1	0

no of resources still needed by each process to proceed

Banker’s algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
4	2	3	2

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms	no of resources held by each process
P1	-	-	-	-	
P2	-	-	-	-	
P3	-	-	-	-	
P4	-	-	-	-	
P5	2	1	1	0	

Process	Tape Drives	Plotters	Scanners	CD Roms	no of resources still needed by each process to proceed
P1	0	0	0	0	
P2	0	0	0	0	
P3	0	0	0	0	
P4	0	0	0	0	
P5	0	0	0	0	

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms	no of resources held by each process
P1	-	-	-	-	
P2	-	-	-	-	
P3	-	-	-	-	
P4	-	-	-	-	
P5	-	-	-	-	

Process	Tape Drives	Plotters	Scanners	CD Roms	no of resources still needed by each process to proceed
P1	0	0	0	0	
P2	0	0	0	0	
P3	0	0	0	0	
P4	0	0	0	0	
P5	0	0	0	0	

Banker's algorithm for multiple resource

Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
1	0	2	0

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	3	0	1	1
P2	0	1	0	0
P3	1	1	1	0
P4	1	1	0	1
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	1	1	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	1	1
P5	2	1	1	0

no of resources still needed by each process to proceed

Deadlock prevention

- Deadlock can be prevented by **attacking the one of the four conditions** that leads to deadlock.
  1. Attacking the **Mutual Exclusion Condition**
     - No deadlock **if each resource can be assigned to more than one process**.
     - We **can not assign some resources to more than one process at a time** such as **CD-Recorder, Printer** etc…
     - So this solution is **not feasible**.
  2. Attacking the **Hold and Wait Condition**
     - Require processes to **request all their resources before starting execution**.
     - A process is **allowed to run if all resources it needed is available**. Otherwise nothing will be allocated and it will just wait.
     - Problem with this strategy is that a **process may not know required resources at start of run**.
     - Resource will **not be used optimally**.
  3. Attacking the **No Preemption Condition**
     - When a process P0 request some resource R which is held by another process P1 then resource R is **forcibly taken away from the process P1 and allocated to P0**.
     - Consider a process holds the **printer**, halfway through its job; **taking the printer away from this process without having any ill effect is not possible**.
     - This is not a **possible option**.

Deadlock prevention

- Deadlock can be prevented by **attacking the one of the four conditions** that leads to deadlock.
  4. Attacking the **Circular Wait Condition**
     - Provide a **global numbering of all the resources**.
       1) Printer
       2) Scanner
       3) Plotter
       4) Tape Drive
       5) CD Rom
     - Now the rule is that: **processes can request resources whenever they want to, but all requests must be made in numerical order**.
     - A process **need not acquire them all at once**.
     - Circular wait is prevented if a **process holding resource n cannot wait for resource m, if m > n**.
     - A process **may request 1st a CD ROM, then tape drive**. But it **may not request 1st a tape drive, then CD ROM.**
     - **Resource graph can never have cycle.**

Example

- Consider a system consisting of four resources of same type that are shared by three processes, each of which needs at most two resources. Show the system is deadlock free

Process	Has / Hold	Max required
A	1	2
B	1	2
C	1	2
Total : 4 & Free : 1		

Process	Has / Hold	Max required
A	2	2
B	1	2
C	1	2
Free : 0		

Process	Has / Hold	Max required
A	0	-
B	1	2
C	2	7
Free : 2		

Process	Has / Hold	Max required
A	0	-
B	2	2
C	7	7
Free : 1		

Process	Has / Hold	Max required
A	0	-
B	0	-
C	1	2
Free : 3		

Process	Has / Hold	Max required
A	0	-
B	0	-
C	2	2
Free : 2		

Questions

1. What is RAG? Explain briefly.
2. What is Deadlock? List the conditions that lead to deadlock. How Deadlock can be prevented?
3. Which are the necessary conditions for Deadlock? Explain Deadlock recovery in brief.
4. Consider the snapshot of the syste resources A,B,C,D.
   - Currently available set of resources (1,5,2,0).
   - Answer the following Questions usir bankers algorithm.
     1. Find the content of Need Matrix.
     2. Is the System in Safe State?
     3. If request from Process P1 arrives for (0,4,2,0) can the request be granted immediately

	Allocation					Max			
Process	A	B	C	D		A	B	C	D
P0	0	0	1	2		0	0	1	2
P1	1	0	0	0		1	7	5	0
P2	1	3	5	4		2	3	5	6
P3	0	6	3	2		0	6	5	2
P4	0	0	1	4		0	6	5	6