Deadlock

Prof. Harish D.G.
Dept. of Computer and IT
College of Engineering,Pune
www.harishgadade.com

Deadlock

- What is Deadlock
- Deadlock Characterization
  - Necessary Conditions
  - Resource Allocation Graph
- Methods to handle deadlock
  - Deadlock Prevention
  - Deadlock Avoidance
  - Deadlock Detection and Recovery

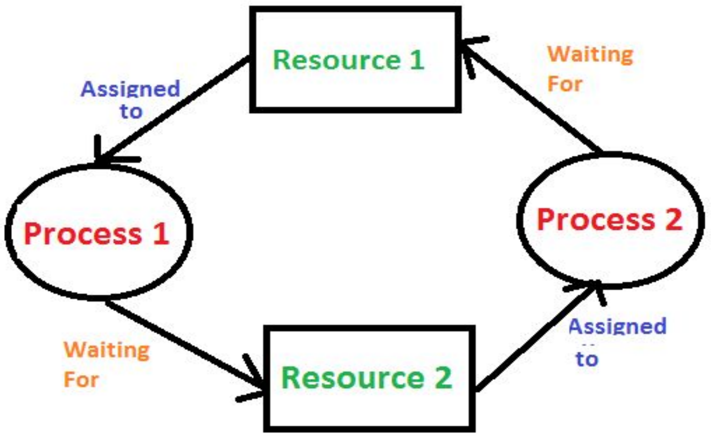
Deadlock

A process in operating system uses resources in the following way.

1) Requests a resource
2) Use the resource
3) Releases the resource

- **What is Deadlock?**

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.



Deadlock Characterization

Before we discuss various methods for dealing with the deadlock problems, we shall describe features that characterize the deadlock

1. Necessary Conditions:

Followings are the necessary and sufficient condition for Deadlock;

1) Mutual Exclusion
2) No Preemption
3) Hold and Wait
4) Circular Wait

Deadlock Characterization

2. Resource-Allocation Graph:

- Deadlock can be described more precisely in terms of Directed Graph called a System Resource Allocation Graph.
- RAG is a set of vertices (V) and edges(E)

Set of vertices are Set of processes and set of resources.

$P = \{ P1, P2, P3, \ldots P_n \}$

$R = \{ R1, R2, R3, \ldots R_n \}$

Set of Edges are Request edge and Assignment Edge

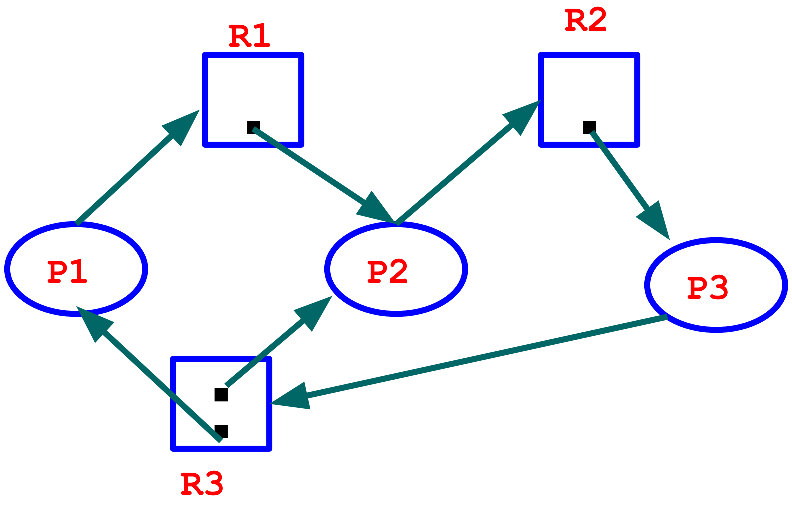
$P_i \rightarrow R_j$

$R_j \rightarrow P_i$

Deadlock Characterization

2. Resource-Allocation Graph:

Example:



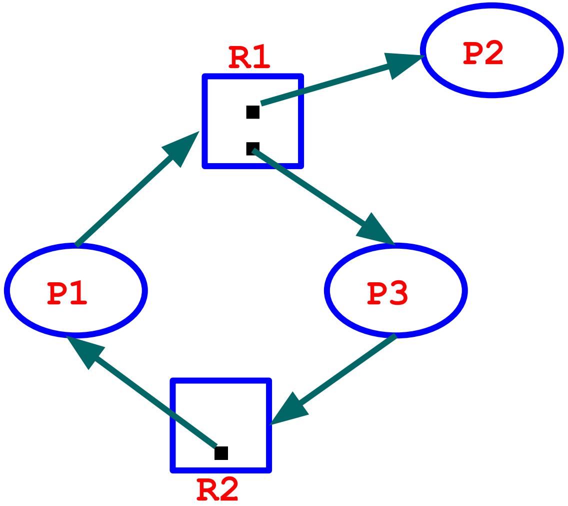
Processes	Allocation R1 R2 R3	Request R1 R2 R3	Available R1 R2 R3
P1	0 0 1	1 0 0	0 0 0
P2	1 0 1	0 1 0	
P3	0 1 0	0 0 1	

This RAG contains Deadlock

Deadlock Characterization

2. Resource-Allocation Graph:

Example:



Processes	Allocation R1 R2	Request R1 R2	Available R1 R2
P1	0 1	1 0	0 0 0
P2	1 0	0 0	
P3	1 0	0 1	

This RAG contains No Deadlock

Deadlock Characterization

2. Resource-Allocation Graph:

Therefore we can conclude that

- If Graph contains no cycle, then No DEADLOCK
- If Graph contain a cycle;
  - If only one instances per resource type, then DEADLOCK
  - If several instance per resource type, then there may have a possibility of DEADLOCK.

Methods to Handle Deadlock

Followings are the various methods to handle deadlock;

1) Deadlock Ignorance
2) Deadlock Prevention
3) Deadlock Avoidance
4) Deadlock Detection and Recovery

1. Deadlock Ignorance

Ignore by adding code to Operating System or simply restart your machine

Methods to Handle Deadlock

2. Deadlock Prevention

Try to find solution before deadlock occurs

Necessary Conditions:

a) Mutual Exclusion

b) No Preemption

c) Hold and Wait

d) Circular aait

- Deadlock prevention says,try to remove or make false all four conditions OR at least try to remove or make false any one of the conditions.

Methods to Handle Deadlock

2. Deadlock Prevention

- To prevent Deadlock-
  - Make mutual Exclusion False
    - By Just sharing resources but it is not possible in some resources like printer as it is non sharable.
  - Try to make No-Preemption False
    - Means Preemption is TRUE, can use Time Quantum Method
  - Make Hold and Wait False
    - Try to do No Hold and Wait
  - Make Circular Wait False
    - To remove Circular wait,just give the numbering to all resources

Methods to Handle Deadlock

3. Deadlock Avoidance

- Simplest and most useful model requires that each process declares maximum number of resources that it may need.
- Deadlock avoidance Algorithm dynamically examines the resources allocation can never be a circular wait condition.
- **Basic Fact:**
  - If a system is in Safe State, No Deadlock
  - If a System is in Unsafe State, Possibility of Deadlock.
- **Avoidance:-** Ensure that a system will never enter in Unsafe State.

Methods to Handle Deadlock

3. Deadlock Avoidance

- Allow the system to enter into deadlock State
- Deadlock Detection Algorithms
    - Single Instance
    - Multiple Instance
- For Single Instance, Wait-for-Graph Algorithm is used
- For Multiple Instance, Banker's Algorithm is used
- In Wait-for-Graph, if cycle exists, then we can say that, there is a Deadlock but in multiple instance, if cycle exists, there may or may not be a Deadlock.

Methods to Handle Deadlock

3. Deadlock Avoidance

A. Safe State

Example: Suppose there as four processes in execution with 12 instances of a resource R in a system.

Processes	Max Need	Current Allocation
P1	8	3
P2	9	4
P3	5	2
P4	3	1

With the reference to current allocation, Is system Safe? If so, what is the safe state sequence.

Methods to Handle Deadlock

3. Deadlock Avoidance

- **Example: Deadlock Avoidance Using Banker,s Algorithm**

  Total resources are A=10,B=5, C=7 and five processes with following need. Find safe sequence to avoid deadlock.

Processes	Allocation			Maximum			Current Work(Available)			Remaining Need(Max - Alloc)		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2	7	4	3
P1	2	0	0	3	2	2	5	3	2	1	2	2
P2	3	0	2	9	0	2	7	4	3	6	0	0
P3	2	1	1	4	2	2	7	4	5	2	1	1
P4	0	0	2	5	3	3	7	5	5	5	3	1
	7	2	5				10	5	7			

Methods to Handle Deadlock

3. Deadlock Avoidance

Current Work = Total - Total Allocation

For A, CW = 10 - 7 = 3

For B, CW = 5 - 2 =3

For C, CW = 7 - 2 = 2

I.e. (A B C) = (3 3 2)

Banker's Algorithm(Deadlock Avoidance/Detection)

Needi <= Work, Work = Work + Allocation

P0, 7 4 3 <= 3 3 2, Not TRUE
P1, 1 2 2 <= 3 3 2, TRUE, then W = 3 3 2 + 2 0 0 = 5 3 2
P2, 6 0 0 <= 5 3 2, Not TRUE
P3, 2 1 1 <= 5 3 2, TRUE, then W = 5 3 2 + 2 1 1 = 7 4 3

Methods to Handle Deadlock

3. Deadlock Avoidance

P4, 5 3 1 <= 7 4 3, TRUE, then W = 7 4 3 + 0 0 2 = 7 4 5
P0, 7 4 3 <= 7 4 5, TRUE, then W = 7 4 5 + 0 1 0 = 7 5 5
P2, 6 0 0 <= 7 5 5, TRUE, then W = 7 5 5 + 3 0 2 = 10 5 7

Thus,

The Safe Sequence is = P1, P3, P4, P0,P2

Therefor, in this sequence, deadlock will not occur.

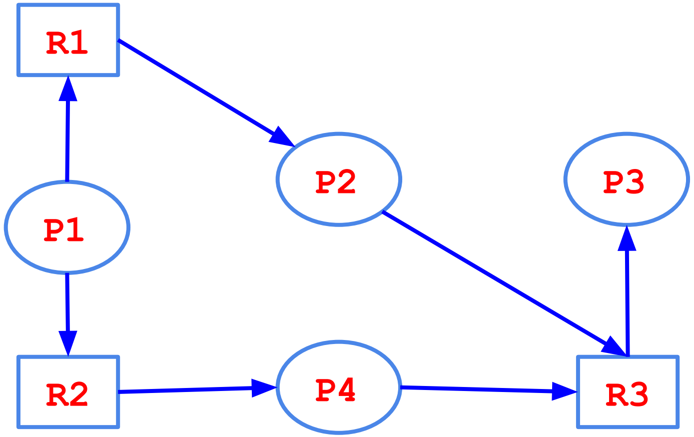
Methods to Handle Deadlock

4. Deadlock Detection and Recovery

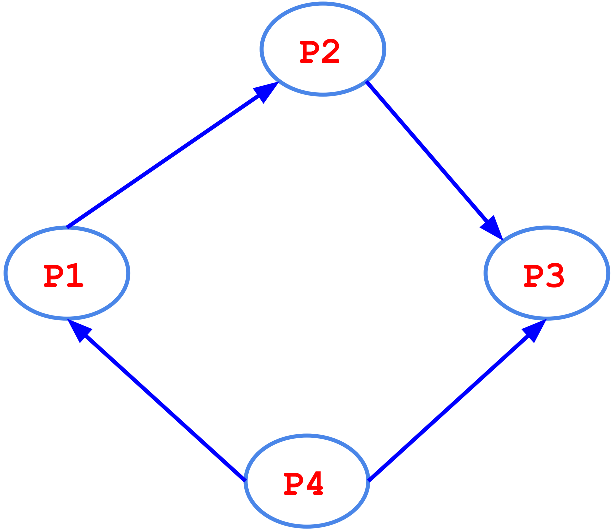
- Allow the system to enter into deadlock State
- Deadlock Detection Algorithms
  - Single Instance
  - Multiple Instance
- For Single Instance, Wait-for-Graph Algorithm is used
- For Multiple Instance, Banker's Algorithm is used
- In Wait-for-Graph, if cycle exists, then we can say that, there is a Deadlock but in multiple instance, if cycle exists, there may or may not be a Deadlock.

Methods to Handle Deadlock

4. Deadlock Detection and Recovery

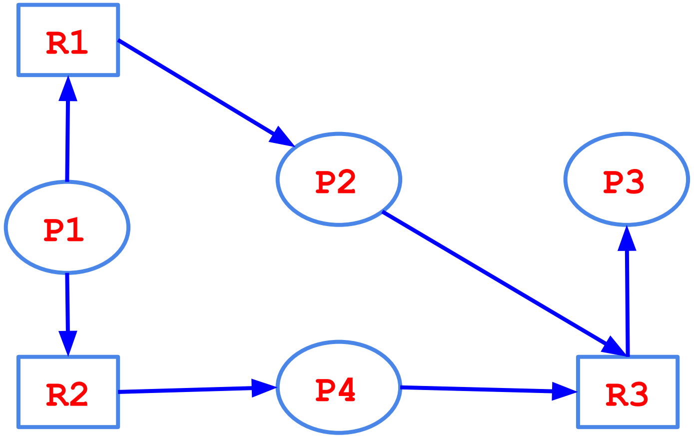


Resource Allocation Graph (RAG)



Methods to Handle Deadlock

4. Deadlock Detection and Recovery



Resource Allocation Graph (RAG)

Processes	Max Need R1 R2 R3	Current Allocation R1 R2 R3
P1	1 1 0	0 0 0
P2	0 0 1	1 0 0
P3	0 0 0	0 0 1
P4	0 0 1	0 1 0