Lesson 15

Objectives

- Deadlock introduction
- Deadlock characterization
- Resource Allocation Graph
- Deadlock and RAG

Deadlock Introduction

A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set.

Example 1

✦ System has 2 tape drives.

✦ P1 and P2 each hold one tape drive and each needs another one.

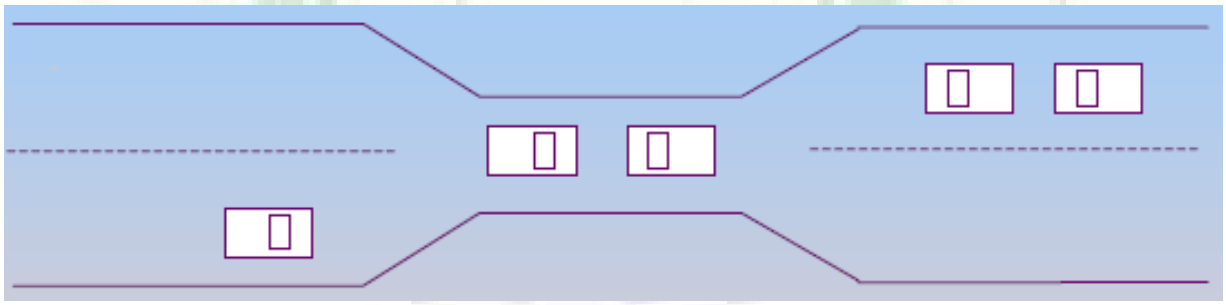
Example 2

✦ Semaphores A and B, initialized to 1

P0	P1
wait (A);	wait(B)
wait (B);	wait(A)

Example 3

A bridge crossing scenario



Traffic only in one direction.

- Each section of a bridge can be viewed as a resource.
- If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback).
- Several cars may have to be backed up if a deadlock occurs.
- Starvation is possible.

System Model

Suppose there is a set of resource types R1, R2, . . ., Rm. For example, CPU cycles, memory space, I/O devices etc.

- Each resource type Ri has Wi instances.
- Each process utilizes a resource as follows:
  - ✦ Request
  - ✦ Use
  - ✦ Release

Deadlock Characterization

There are four necessary and sufficient conditions for the deadlock. Deadlock can arise if and only if four conditions hold simultaneously.

1. **Mutual exclusion**:

   Only one process at a time can use a resource. No two or more processes can use the same resource simultaneously.

   For example: CPU can be utilized by one process at a time.

2. **Hold and wait**:

   A process holding at least one resource is waiting to acquire additional resources held by other processes.

   For example: a process may need a disk drive as well as a printer so such situation is obvious.

3. **No preemption**:

   A resource can be released only voluntarily by the process holding it, after that process has completed its task. Or a process will not relinquish control of a resource by influence of any other process or whatever.

   For example: A process is running in its critical section, so no other process can preempt the existing process.

4. **Circular wait**:

   There exists a set {P0, P1, …, Pm} of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2, …, Pn–1 is waiting for a resource that is held by Pn, and Pn is waiting for a resource that is held by P0.

Resource Allocation Graph

A graph is a function of two variables namely *Set of Vertices* and *Set of Edges*. That is, $G = f(V, E)$. Function specifies that which edge connects which pair of vertices etc. There are two kinds of vertices and two kinds of edges in the Resource Allocation Graph. So set V is partitioned into two types:

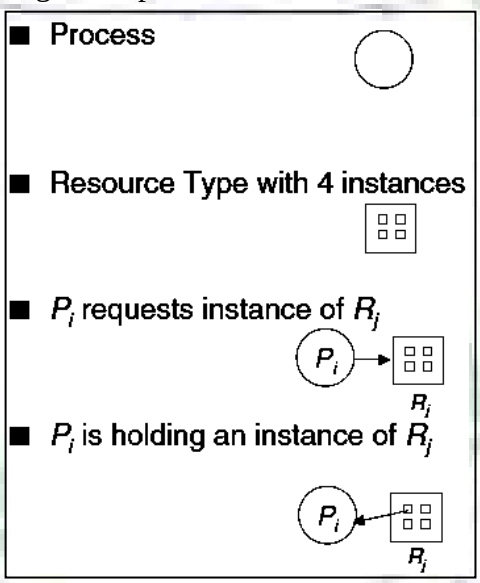
- P = {P1, P2, …, Pn}, the set consisting of all the processes in the system.
- R = {R1, R2, …, Rm}, the set consisting of all resource types in the system.

Similarly, set E is partitioned into two types

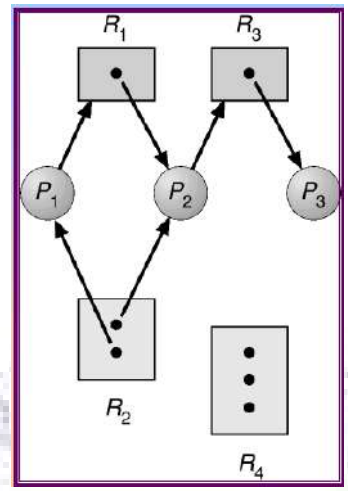
- Request edge – directed edge P1 → Rj

  Means process P1 is requesting for resource Rj
- Assignment edge – directed edge Rj → Pi

  Means resource Rj is assigned to process Pi



Example of a Resource Allocation Graph



What information does a Resource Allocation Graph convey?

- How many processes are there in the system
- How many resources are there in the system
- How many instances per resource type are there in the system
- Which process is requesting for which resource/s and which resource/s are allocated to which process
- How many resources are occupied and how many are unoccupied
- Where there is a **deadlock** in the system

Deadlock and Resource Allocation Graph

- If there is no cycle in Resource Allocation Graph then there is no deadlock
- If there is a cycle in Resource Allocation Graph then there may be a deadlock
    - **Cycle with deadlock** when there is no free running process means all processes are taking part in circular wait and no one is executing
    - **Cycle without deadlock** when there is at least one free running process that is not taking part in circular waiting

Cycle with Deadlock **Cycle without Deadlock**