OPERATING SYSTEMS

Deadlocks

Chandravva Hebbi
Department of Computer Science

OPERATING SYSTEMS

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

OPERATING SYSTEMS

Principles of Deadlocks, Deadlock Characterization

Chandravva Hebbi

Department of Computer Science

Introduction

- Multiprogramming environment: several processes compete to limited number of resources
- A Process is holding a resource(R1) and is waiting for the resources(R2).
- The resource R2 is held by another process..
- Waiting state of processes will not change, as the requested resource is held by the waiting process.
- This situation is called deadlock

System Model

- System consists of finite number of resources
- Resource types $R_1, R_2, \ldots, R_m$
  - *Physical resources: CPU cycles, memory space, I/O devices, printer, tape drives.*
  - *Logical resources: semaphores, mutex locks, files.*
- Each resource type R_i has W_i instances.
- Ex: if the system has 2 CPU's then CPU has 2 instances
- Each process utilizes a resource as follows:
  - **Request :** Process makes request to the resource. Eg. system call like request(), open(), wait(),allocate() etc
  - **Use:** operates on these resources
  - **Release:** process releases the resources. Eg. Using a system call like release(), close(), signal(), free etc.
- Request and release of semaphore, acquire and release of lock on mutex

Example 1

- Consider a system with 3 CD RW drives.
- Suppose 3 processes(p0,p1,p2) are holding one drive each.
- What happens,
  - If a process p0 makes a request for one more drive

Example 2

- Consider a system with one printer one DVD drive.
- Process Pi is holding printer and process Pj is holding DVD drive.
- What happens if
  - Process Pi request DVD and Pj requests printer

Does dead lock occur in example 1 and 2?

Deadlock with Semaphores

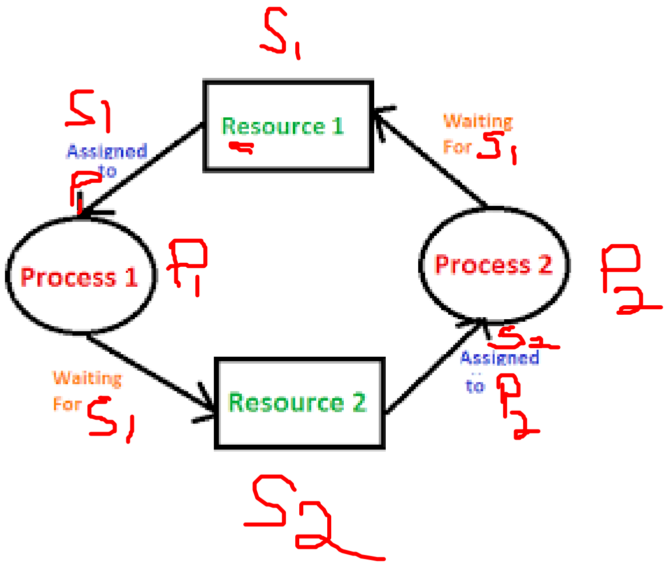
- Data:
  - A semaphore **S1** initialized to 1
  - A semaphore **S2** initialized to 1
- Two processes P1 and P2
- **P1:**

  wait(s1)

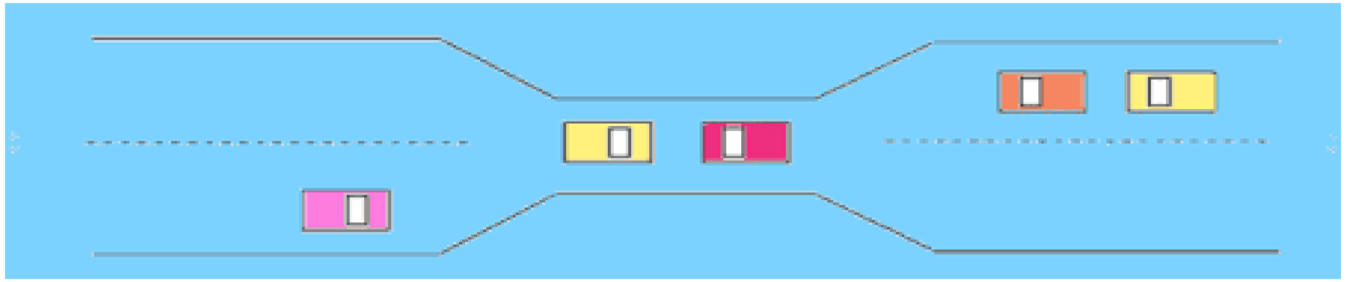
  wait(s2)
- **P2:**

  wait(s2)

  wait(s1)



Bridge crossing Example



- 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.
- Note – Most OSes do not prevent or deal with deadlocks

Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously.

- **Mutual exclusion**: only one process at a time can use a resource (sharable resources like Read-only files do not require mutually exclusive access and thus cannot be involved in a deadlock.
- **Hold and wait**: a process holding at least one resource is waiting to acquire additional resources held by other processes
- **No preemption**: a resource can be released only voluntarily by the process holding it, after that process has completed its task
- **Circular wait**: there exists a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, …, P_{n-1} is waiting for a resource that is held by P_n, and P_n is waiting for a resource that is held by P_0.

Resource-Allocation Graph

- Deadlocks are described precisely with directed graphs called system resource-allocation graph

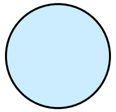
A graph consists of set of vertices *V* and a set of edges *E*.

- V is partitioned into two types:
  - $P = \{P_1, P_2, \ldots, P_n\}$, the set consisting of all the processes in the system
  - $R = \{R_1, R_2, \ldots, R_m\}$, the set consisting of all resource types in the system
- **request edge** – directed edge $P_i \rightarrow R_j$
- **assignment edge** – directed edge $R_j \rightarrow P_i$

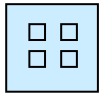
Resource-Allocation Graph (Cont.)

A set of vertices *V* and a set of edges *E*.

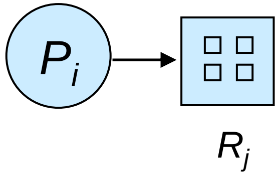
- Process



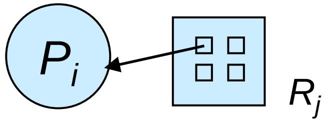
- Resource Type with 4 instances



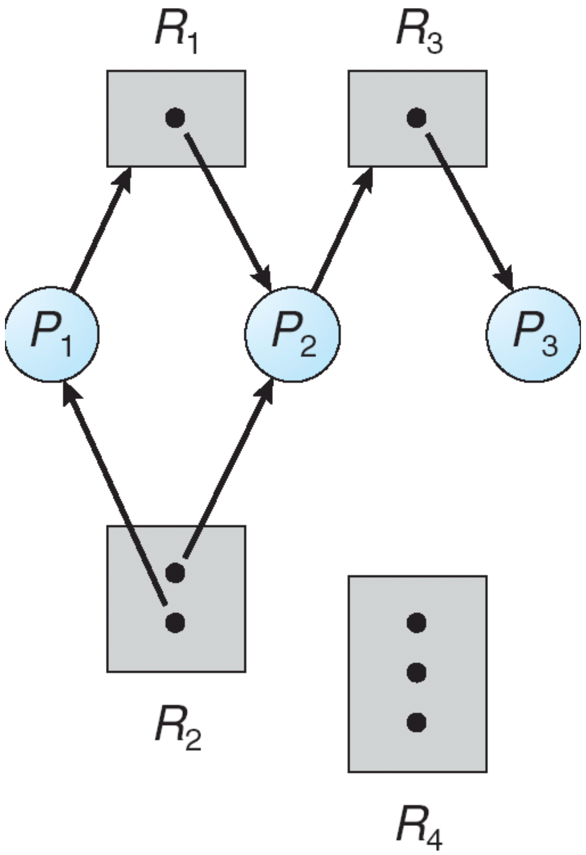
- P_i requests instance of R_j



- P_i is holding an instance of R_j



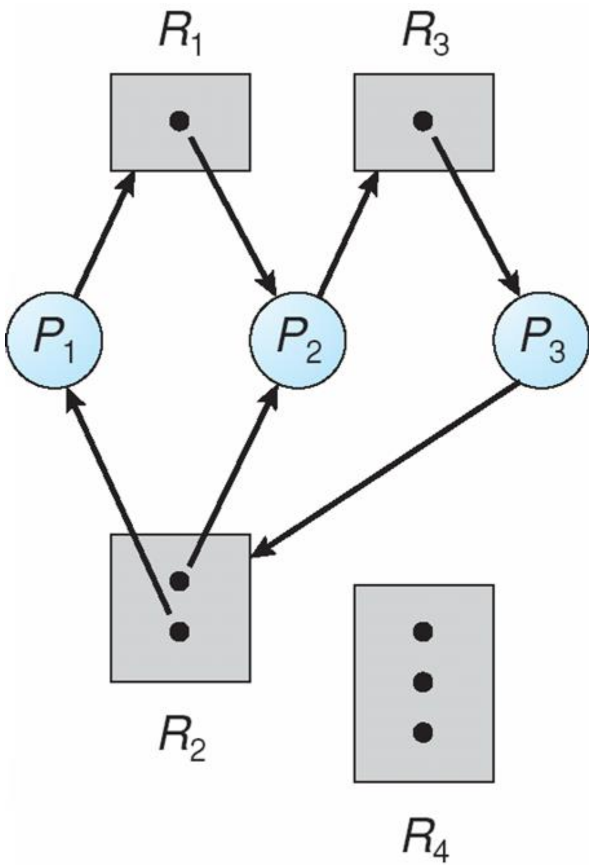
Example of a Resource-Allocation Graph



Set of process: P1,P2,P3

Set of Resources: R1,R2,R3

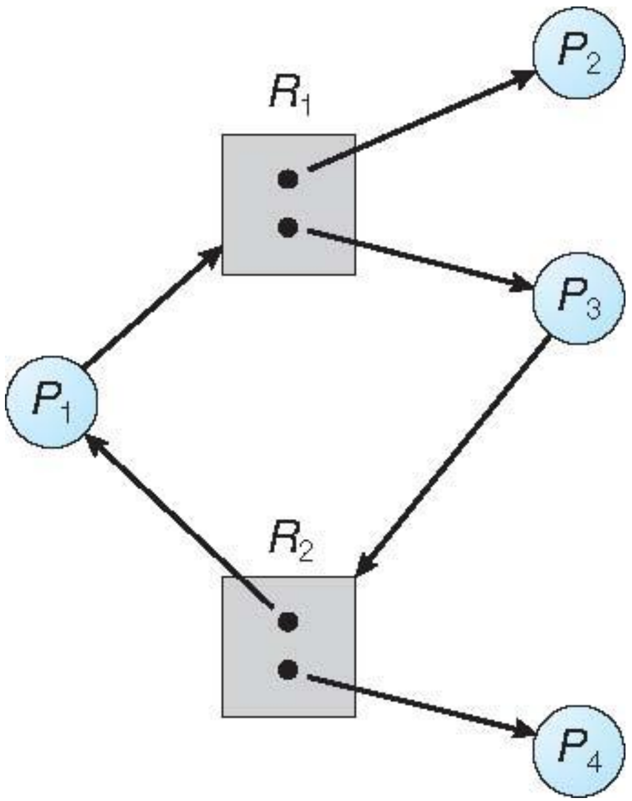
Resource-Allocation Graph With A Deadlock



P1→R1→ P2→R3→ P3→R2→P1

P2→R3→P3→R2→P2

Resource-Allocation Graph With A Cycle but No Deadlock



P1→R1→P3→R2→P1

Basic Facts

- If graph contains no cycles $\Rightarrow$ no deadlock
- If graph contains a cycle $\Rightarrow$
  - if only one instance per resource type, then deadlock
  - if several instances per resource type, the system may or may not be in a deadlocked state

PES UNIVERSITY ONLINE

THANK YOU

Chandravva Hebbi
Department of Computer Science Engineering
chandravvahebbi@pes.edu