Deadlock (I)

The Deadlock Problem

- A set of blocked processes each holds an instance of resource and waits to acquire an instance of resource held by another process
- Example
  - System has 2 disk drives
  - P_1 and P_2 each hold one disk drive and each needs another one
- Example
  - semaphores A and B, initialized to 1

P_0	P_1
wait (A);	wait(B)
wait (B);	wait(A)

Deadlock Characterization

Deadlock arises → 4 conditions hold simultaneously
(necessary conditions of deadlock)

- **Mutual exclusion:** an instance of resource can only be used by a process at a time
- **Hold and wait:** a process holding at least one instance of resource is waiting to acquire additional resources held by other processes
- **No preemption:** 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 $P_2, \ldots, P_{n-1}$ 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.

System Model

- Resource types $R_1, R_2, \ldots, R_m$

  CPU cycles, memory space, I/O devices

- Each resource type R_i has W_i instances.
- Each process utilizes a resource in the following order:

  request → use → release

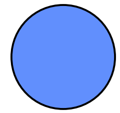
Resource-Allocation Graph

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

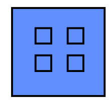
- V is partitioned into two types:
  - $P = \{P_1, P_2, \ldots, P_n\}$, the set consisting of all the processes in the system
  - $R = \{R_1, R_2, \ldots, R_m\}$, the set consisting of all resource types in the system
- request edge – directed edge $P_i \rightarrow R_j$
- assignment edge – directed edge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

Resource-Allocation Graph (Cont.)

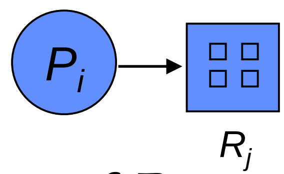
- Process



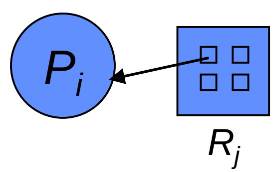
- Resource Type with 4 instances



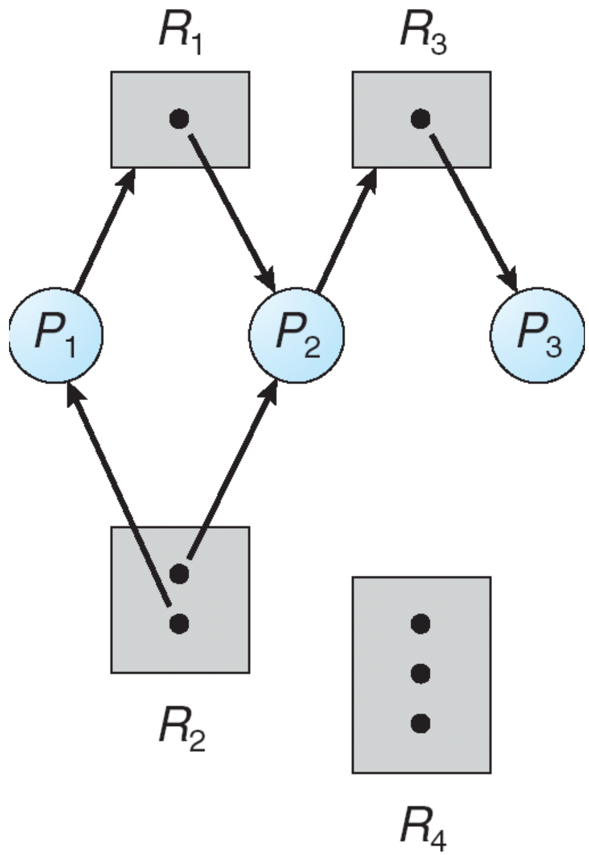
- P_i requests an instance of R_j



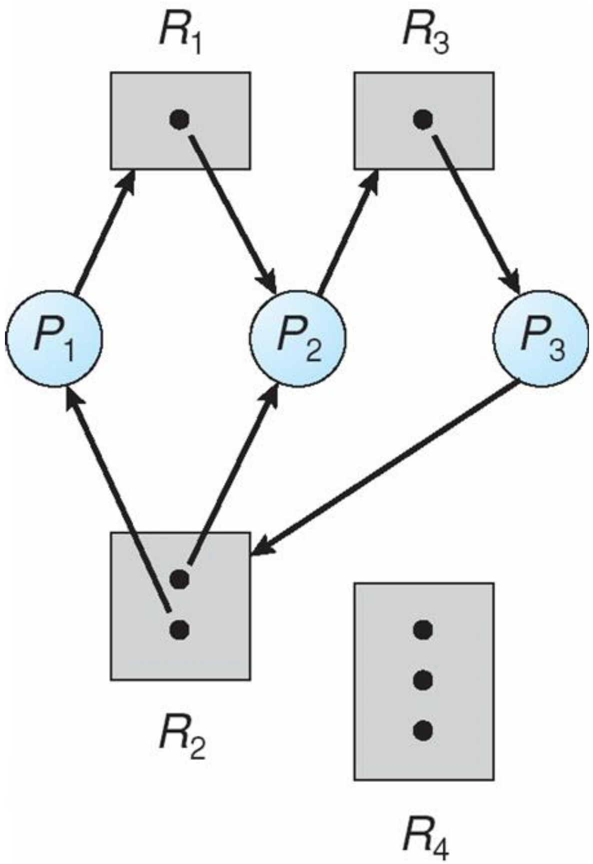
- P_i is holding an instance of R_j



Example of a Resource Allocation Graph

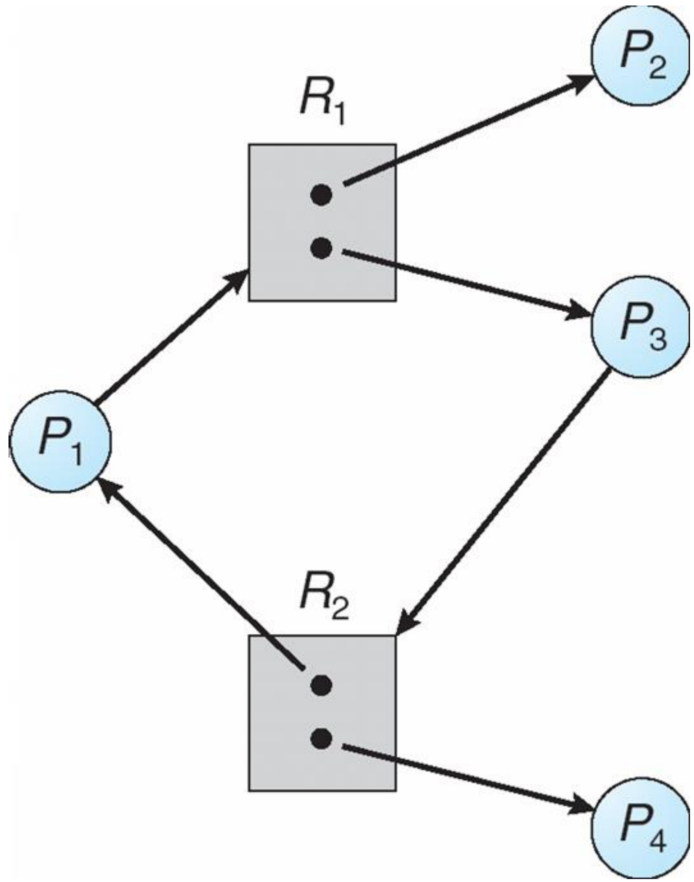


Resource Allocation Graph With A Deadlock



There is one or more cycles in the graph

Graph With A Cycle But No Deadlock



Basic Facts

- If graph contains no cycles ⇒ no deadlock
- If graph contains a cycle ⇒
  - if only one instance per resource type, then deadlock
  - if several instances per resource type, possibility of deadlock

Methods for Handling Deadlocks

- Ensure that the system will *never* enter a deadlock state
- Allow the system to enter a deadlock state and then recover
- Ignore the problem and pretend that deadlocks never occur in the system; used by most operating systems, including UNIX

Deadlock Prevention

Restrain the ways that request can be made such that at least one of the deadlock necessary conditions fails

- **Mutual Exclusion → Make each instance of resource simultaneously sharing** – often impossible

- **Hold and Wait → Request all at once; or, Give up all before request** – must guarantee that whenever a process requests a resource, it does not hold any other resources (i.e., it should release all resources that it currently holds)
  - Low resource utilization; starvation possible

Deadlock Prevention

- **No Preemption → Allow resource instance to be re-assignable**
  - Often impossible
- **Circular Wait → Prevent Circular Wait** – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration

Deadlock due to circular wait

P1 & P2 share semaphores A & B & C

P1:	P2:
wait(A)	wait(B)
...segment 1…	...segment 4…
wait(B)	wait(C)
...segment 2…	...segment 5…
wait(C)	wait(A)
…segment 3 …	…segment 6 …
signal(C)	signal(A)
signal(B)	signal(C)
signal(A)	signal(B)

Prevent circular wait

P1 & P2 share semaphores A & B & C; impose order: A<B<C

P1:

wait(A)

...segment 1…

wait(B)

...segment 2…

wait(C)

…segment 3 …

signal(C)

signal(B)

signal(A)

P2:

wait(A)

wait(B)

...segment 4…

wait(C)

...segment 5…

…segment 6 …

signal(A)

signal(C)

signal(B)