CS 4410
Operating Systems

Deadlocks: Characterisation & Prevention

Summer 2011

Cornell University

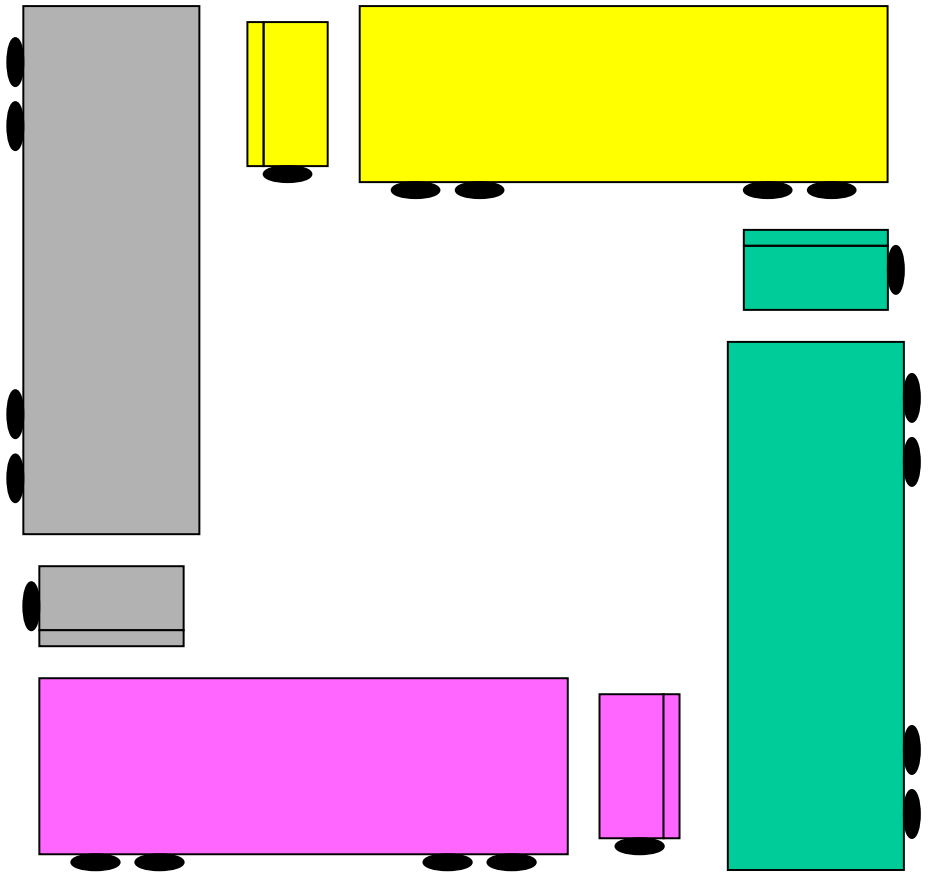
Today

- What are the deadlocks and how are they created?
- System Model
- Deadlock examples
- Deadlock
- Four conditions for deadlock
- Resource allocation graph
- Deadlock prevention

System Model

- There are **non-shared** computer resources
  - Maybe more than one instance
  - Printers, Semaphores, Tape drives, CPU
- Processes need **access** to these resources
  - **Acquire** resource
    - If resource is available, access is granted
    - If not available, the process is blocked
  - **Use** resource
  - **Release** resource
- Undesirable scenario:
  - Process A acquires resource 1, and is waiting for resource 2
  - Process B acquires resource 2, and is waiting for resource 1
  - Deadlock!

Deadlock



Deadlock

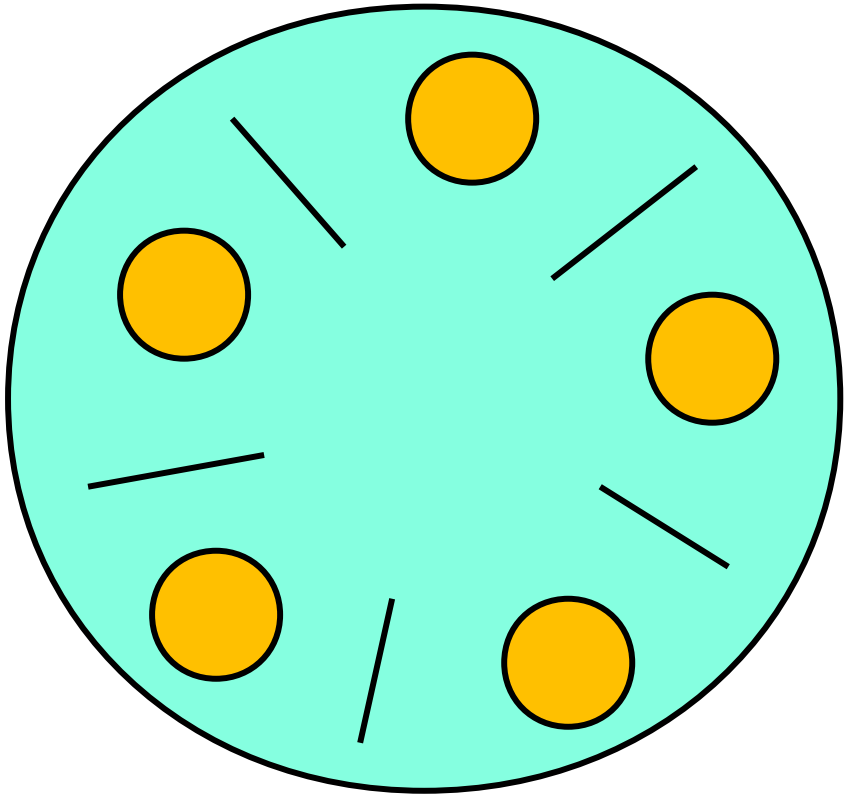
Example 1: Semaphores

```
semaphore:      mutex1 = 1    /* protects file */
                mutex2 = 1    /* protects printer */
```

```
Process A code:
 {
    /* initial compute */
   P(mutex1)
   P(mutex2)

  /* use file & printer*/

   V(mutex2)
   V(mutex1)
}
```

```
Process B code:
 {
    /* initial compute */
   P(mutex2)
   P(mutex1)

  /* use file & printer */

   V(mutex1)
   V(mutex2)
}
```

Example 2: Dining Philosophers



```
class Philosopher:
    chopsticks[N] = [Semaphore(1),…]
    Def __init__(mynum)
        self.id = mynum
    Def eat():
        right = (self.id+1) % N
        left = (self.id-1+N) % N
        while True:
            P(left)
            P(right)
            # eat
            V(right)
            V(left)
```

Deadlock

- A **set of processes** is in a **deadlock** state when **every** process in the set is **waiting** for an **event** that can be **caused** only by **another** process in the set.
- Events: resource acquisition and resource release
- Resources: physical or logical

Four Conditions for Deadlock

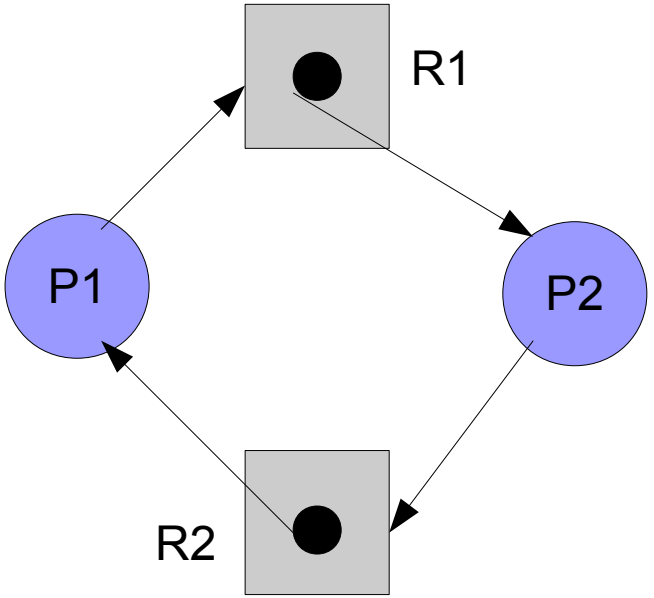
- Necessary conditions for deadlock to exist:
- **Mutual Exclusion**
  - At least one resource must be held in non-sharable mode
- **Hold and wait**
  - There exists a process holding a resource, and waiting for another
- **No preemption**
  - Resources cannot be preempted
- **Circular wait**
  - There exists a set of processes {P1, P2, … PN}, such that
    - P1 is waiting for P2, P2 for P3, …. and PN for P1
- ***All*** four conditions must hold for deadlock to occur

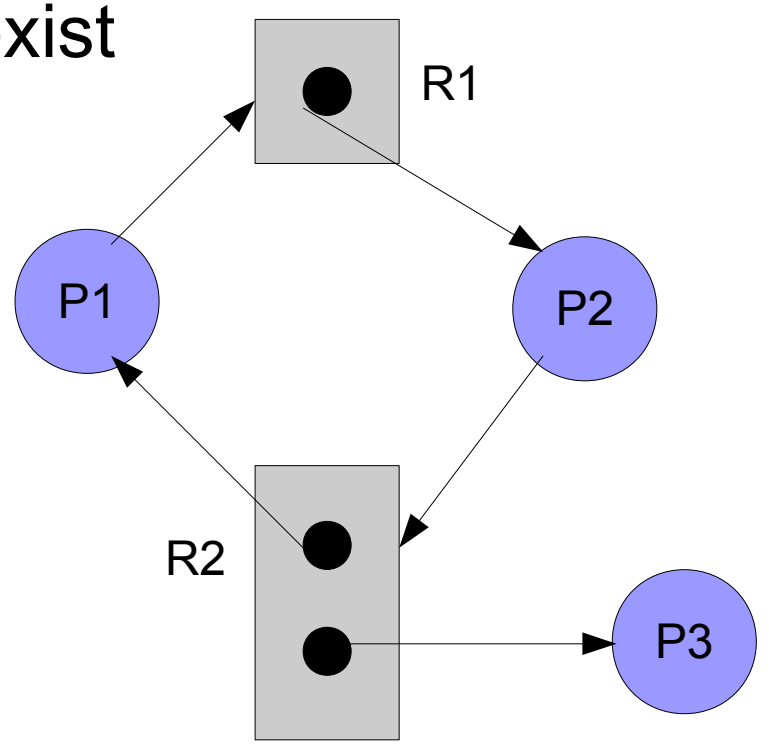
Resource-Allocation Graph

- It helps us **depict** which **resources** have been assigned to which **processes** and which processes have requested which resources.
- Directed graph
- Vertices
  - P: set of processes
  - R: set of resources
- Edges
  - Pi → Rj : request edge
  - Rj → Pi : assignment edge

Resource-Allocation Graph

- If there is a deadlock, then there is a cycle.
- If there is a cycle, then:
  - If the involved resources have one instance each, then there is deadlock
  - Else a deadlock may not exist





Handling Deadlocks

- A system never enters a deadlock state.
  - Prevention, or
  - Avoidance
- A system may enter a deadlock state.
  - Detect deadlock
  - Recover from deadlock
- Ignore deadlock problem

Deadlock Prevention

- Prevention: **Negate one of necessary conditions**
- Mutual exclusion:
  - Make resources sharable
  - Not always possible (printers?)
- Hold and wait
  - Do not hold resources when waiting for another
  - Request all resources before beginning execution
  - Processes do not know what all they will need
  - Starvation (if waiting on many popular resources)
  - Low utilization (Need resource only for a bit)
  - Alternative: Release all resources before requesting anything new
    - Still has the last two problems

Deadlock Prevention

- No preemption:
  - Make resources preemptable (2 approaches)
    - Preempt requesting processes' resources if all not available
    - Preempt resources of waiting processes to satisfy request
  - Good when easy to save and restore state of resource
    - CPU registers, memory virtualization
- Circular wait: (2 approaches)
  - Single lock for entire system? (Problems)
  - Impose **partial ordering** on resources, request them in order

Today

- What are the deadlocks and how are they created?
- System Model
- Deadlock examples
- Deadlock
- Four conditions for deadlock
- Resource allocation graph
- Deadlock prevention