Lecture 09 Deadlock

Dr. Tushar, Mosaddek Hossain Kamal
Professor

Computer Science and Engineering, University of Dhaka,

BSc Third Year, Semester 2 (July – Dec), Academic Year: 2022

CSE3201: Operating Systems

Oct 23, 2022

Outline

Resources

- Examples of Computer Resources
  - printers
  - tape drives
  - Table in a database
- Process needs to access resources in resonable order
- Suppose a process holds resource A and requests resource B
  - at same time another process holds B and requests A
  - both are blocked and remain so
- **Deadlocks occur when** $\cdots$
  - processes are granted exclusive access to devices
  - we refer to these devices generally as resources
- **Preemptable resources**
  - can be taken away from a process with no ill effects
- **Nonpreemptable resources**
  - will cause the process to fail if taken away

Resources

- Sequence of events required to use a resource
  1. request the resource
  2. use the resource
  3. release the resource
- Must wait if request is denied
  - requesting process may be blocked
  - may fail with error code

Example of Resource Usage

```
semaphore res_1, res_2;
void proc_A() {
   down(&res_1);
   down(&res_2);
   use_both_res();
   up(&res_2);
   up(&res_1);
}
void proc_B() {
   down(&res_1);
   down(&res_2);
   use_both_res();
   up(&res_2);
   up(&res_1);
}
```

```
semaphore res_1, res_2;
void proc_A() {
   down(&res_1);
   down(&res_2);
   use_both_res();
   up(&res_2);
   up(&res_1);
}
void proc_B() {
   down(&res_2);
   down(&res_1);
   use_both_res();
   up(&res_1);
   up(&res_2);
}
```

Introduction to Deadlock

- Formal definition

Definition

A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause

- Usually the event is release of a currently held resource
- None of the processes can $\cdots$
  - Run
  - release resources
  - be awakended

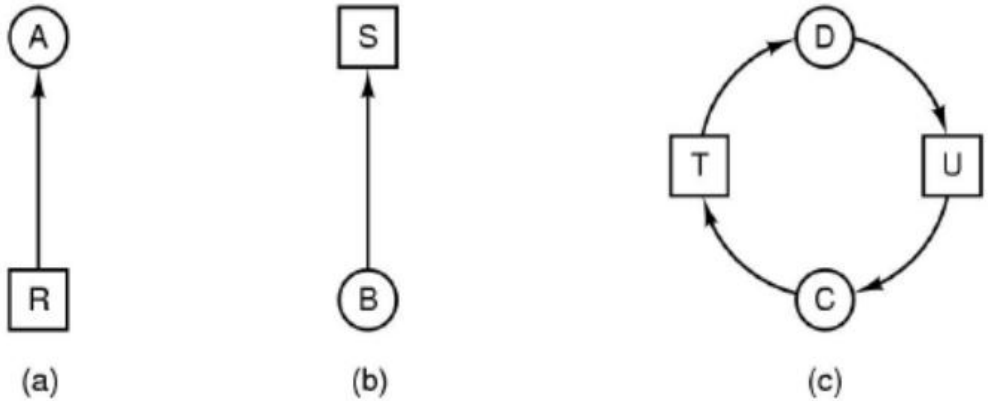
Four Condition for Deadlock

Conditions for deadlock

- Mutual exclusion condition
  - each resource assigned to 1 process or is available
- Hold and wait condition
  - process holding resources can request additional
- No preemption condition
  - previously granted resources cannot forcibly taken away
- Circular wait condition
  - must be a circular chain of 2 or more processes
  - each is waiting for resource held by next member of the chain

Deadlock Modeling

- Modeling with directed graph



  - resource R assigned to process A
  - process B is requesting/waiting for resource S
  - process C and D are in deadlock over resources T and U

Deadlock

- Strategies for dealing with Deadlocks
  1. just ignore the problem altogether
     - Assume there is no deadlock
  2. detection and recovery
     - Let the dealock be occur, we will take necessary actions to recover
  3. dynamic avoidance
     - careful resource allocation
  4. Prevention
     - negating one of the four necessary conditions

Deadlock Modeling

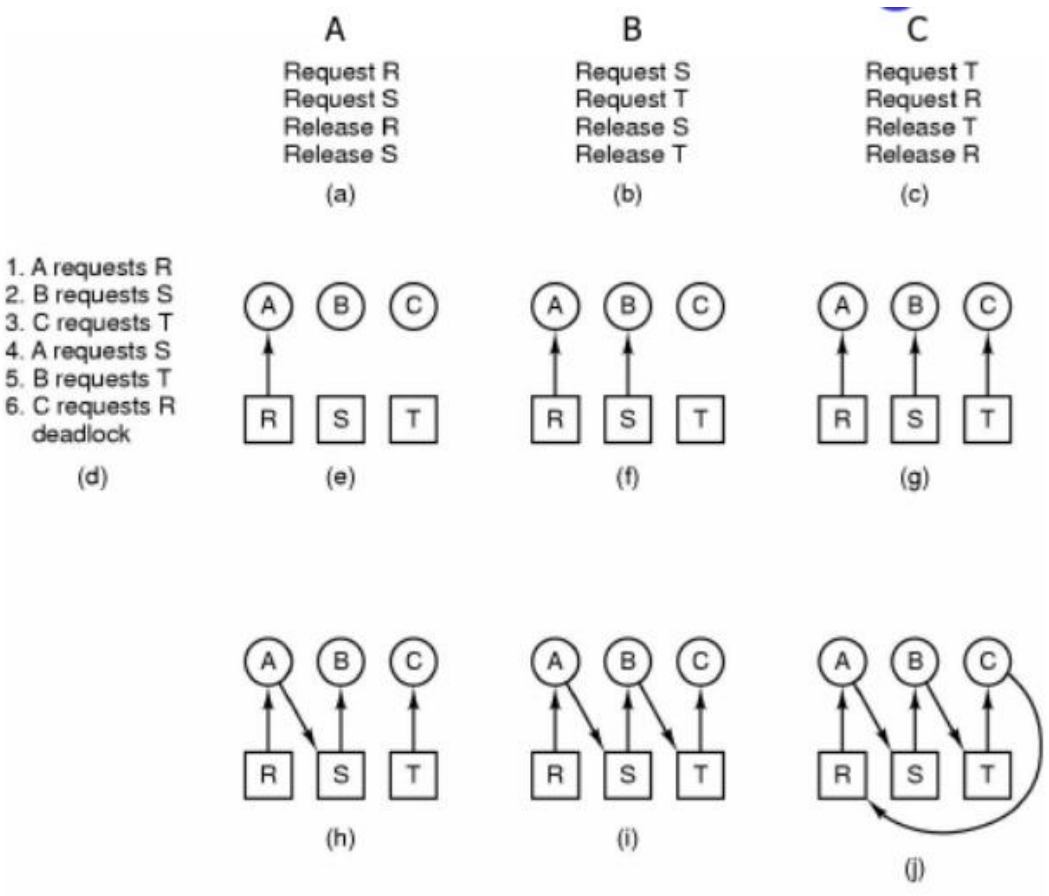


Figure 1: How Deadloack Occurs

Deadlock Modeling

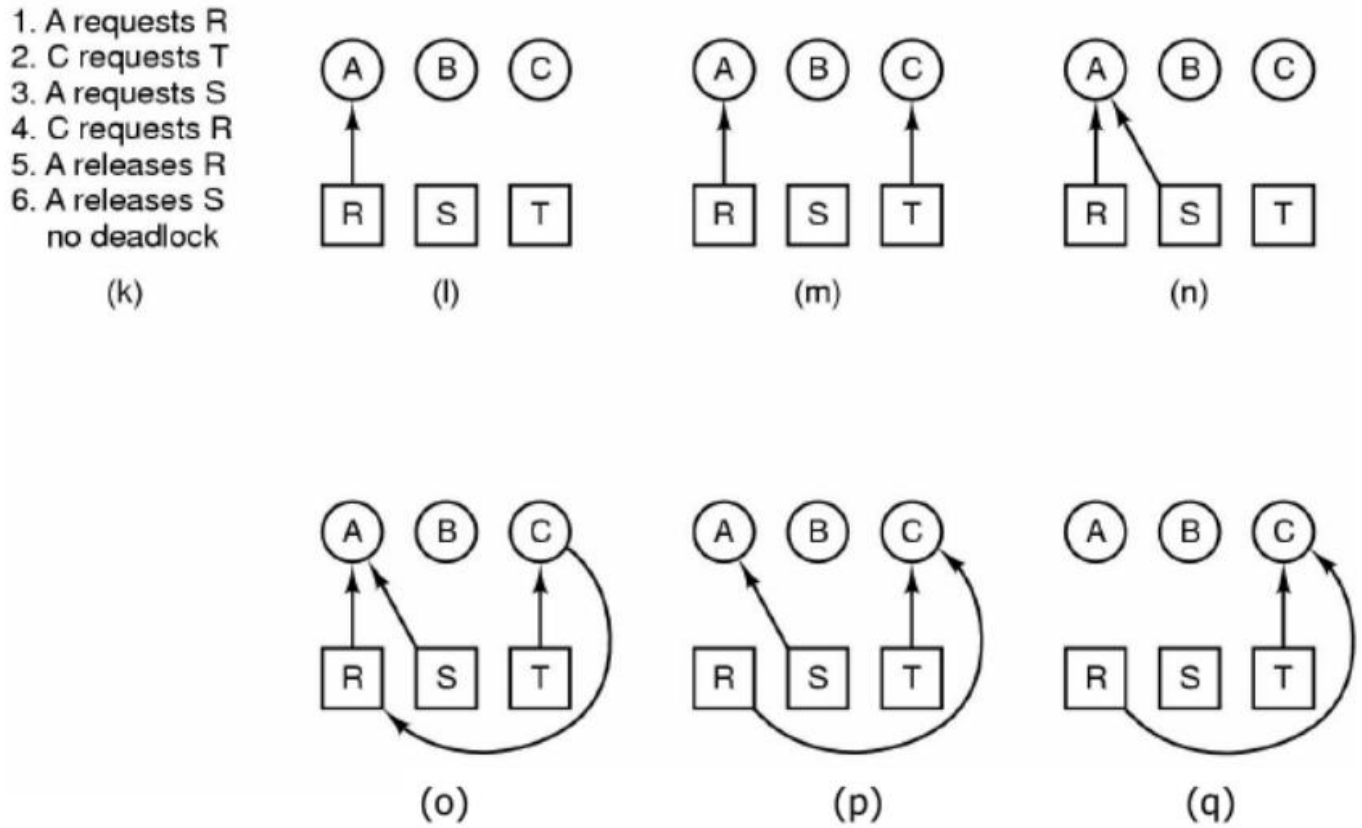
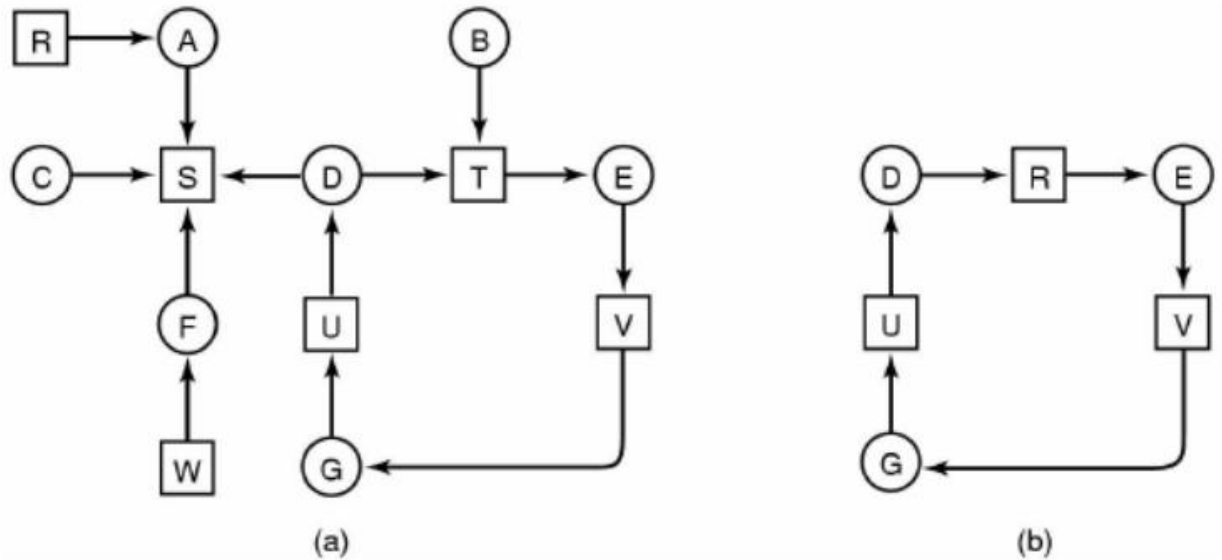


Figure 2: How Deadloack can be Avoided

Approach 1: The Ostrich Algorithm

- Pretend there is no problem
- Reasonable if
  - deadlock occur very rarely
  - cost of prevention is high
    - Example of “cost”, only oneprocess runs at a time
- UNIX and Windows takes this approach
- It’s a tradeoff between
  - Convenience (engineering approach)
  - Correctness (mathematical approach)

Approach 2: Detection with One Resource of Each Type



- Note the resource ownership and requests
- A cycle can be found within the graph, denoting deadlock

However, it is not suitable for detecting deadlock with resource having multiple units.

Resources with multiple instances

Deadlock detection with multiple resources of each type

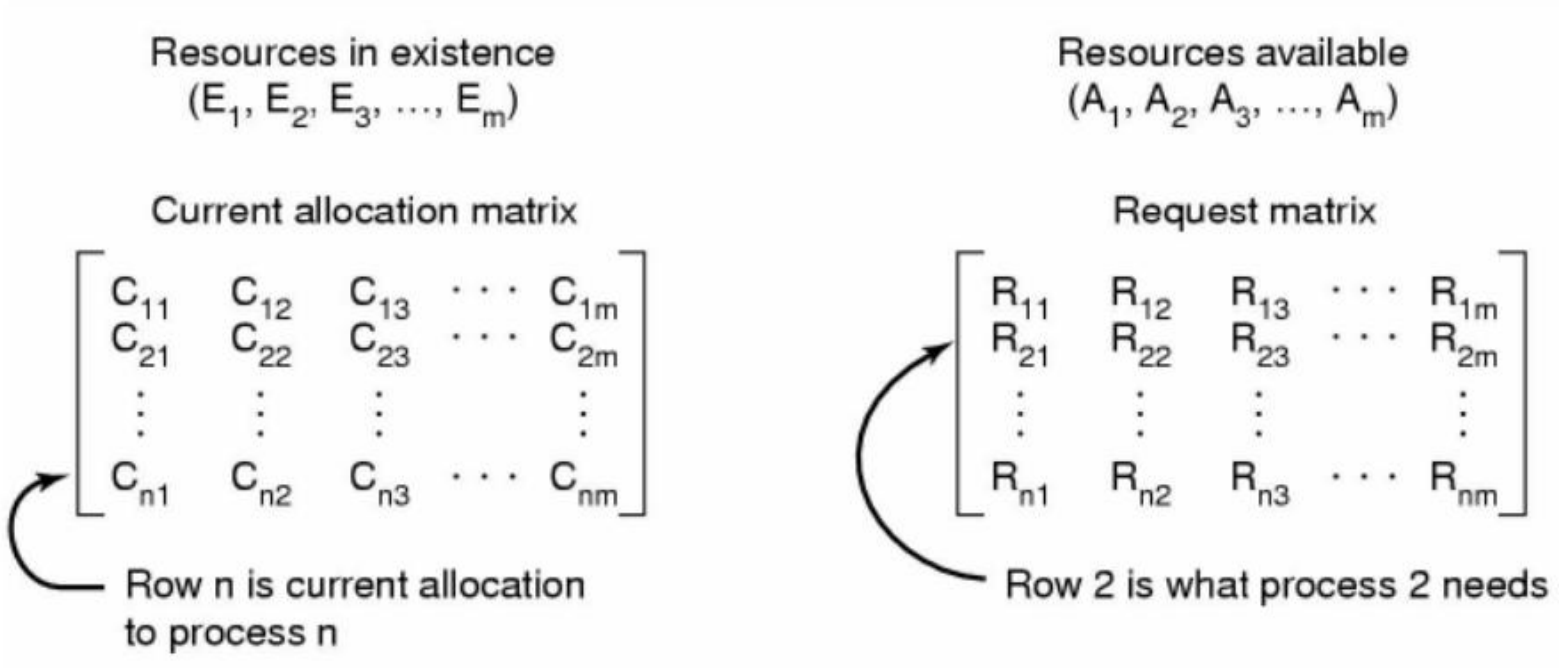


Figure 3: Data Structure needed by dealock detection algorithm

Note the following invariant

Sum of current resource allocation + resource available

= resource that exist

$$\sum_{i=1}^{n} C_{ij} + A_j = E_j$$

Detection with Multiple Resources of Each Type

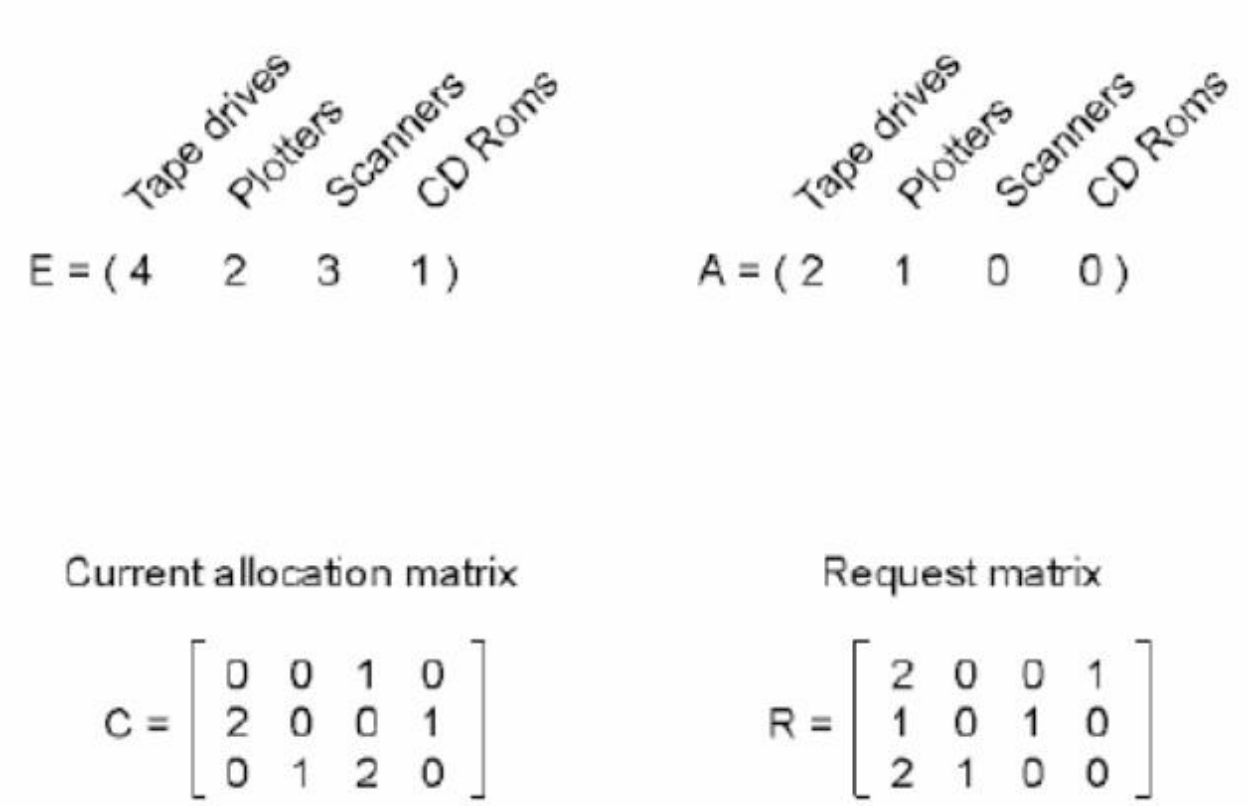


Figure 4: An example for the dealock detection algorithm

Deadlock Detection Algorithm

1. Look for an unmarked process Pi, for which the i-th row of R is less than or equal to A
2. If found, add the i-th row of C to A, and mark Pi. Go to step 1
3. If no such process exists, terminate.

Remaining porcesses are deadlocked.

Example Deadlock Detection

$$E = (4 \quad 2 \quad 3 \quad 1) \qquad A = (2 \quad 1 \quad 0 \quad 0)$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Example Deadlock Detection

purple color use to select a candidate for marking

Blue color use for marking

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Example Deadlock Detection

purple color use to select a candidate for marking

Blue color use for marking

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 2 & 2 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Example Deadlock Detection

purple color use to select a candidate for marking

Blue color use for marking

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 2 & 2 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Example Deadlock Detection

purple color use to select a candidate for marking

Blue color use for marking

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 4 & 2 & 2 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Example Deadlock Detection

purple color use to select a candidate for marking

Blue color use for marking

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 4 & 2 & 2 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Example Deadlock Detection

purple color use to select a candidate for marking

Blue color use for marking

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

- Algorithm Termintes with no unmarked process
  - We have no deadlock

Example 2: Deadlock Detection

- Now, **Suppose P3 needs a CD-ROM as well as 2 Tapes and a Plotter**

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix}$$

Process P1, P2, and P3 in deadlock.

Recovery from Deadlock

- Recovery through preemption
  - take a resource from some other process
  - depends on nature of the resource
- Recovery through rollback
  - checkpoint a process periodically
  - use this saved state
  - restart the process if it is found deadlocked
- Recovery through killing processes
  - crudest but simplest way to break a deadlock
  - kill one of the processes in the deadlock cycle
  - the other processes get its resources
  - choose process that can be rerun from the beginning

Approach 3: Deadlock Avoidance

- Instead of detecting deadlock, can we simply avoid it?
  - YES, but only if enough information is available in advance.
    - Maximum number of each resource required

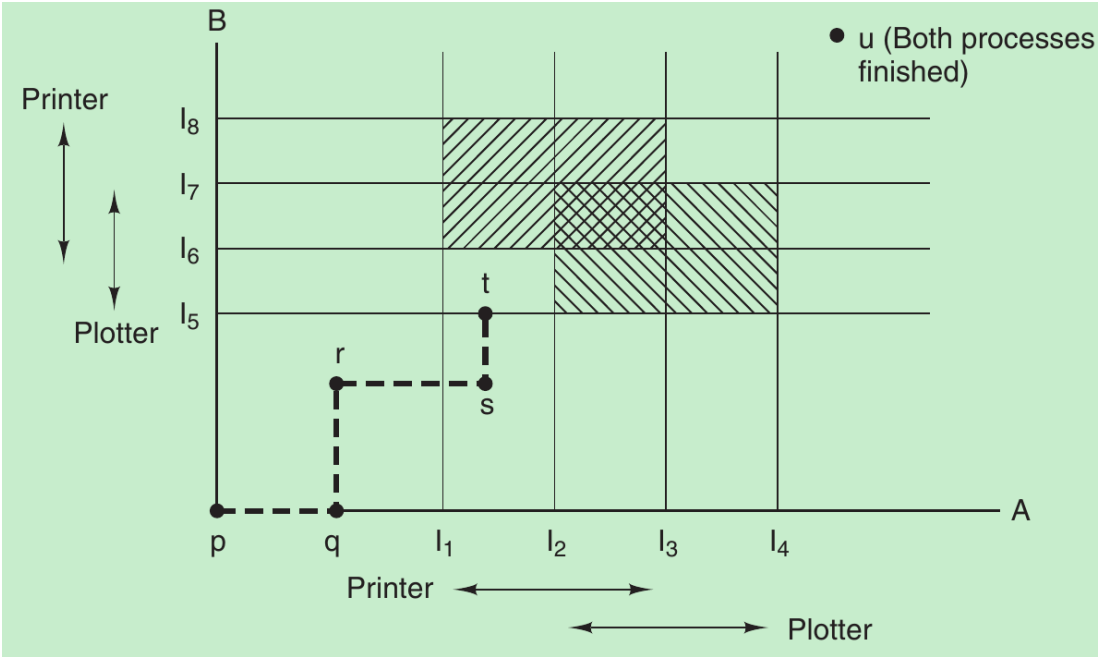
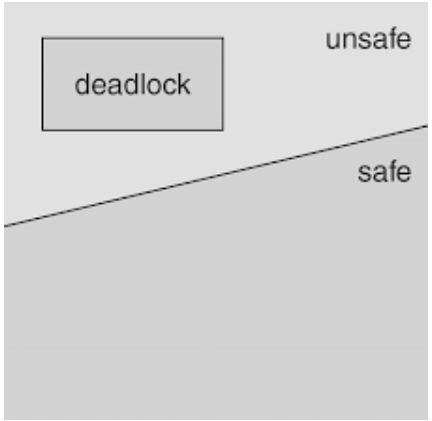


Figure 5: Two process resource trajectories

Safe and Unsafe States

A small subset in unsafe state!!



Safe and Unsafe state of a system

- A system state is safe if
  - The system is not deadlocked
  - There exists a scheduling order that results in every process running to completion, even if they all request their maximum resources immediately

Safe and Unsafe State

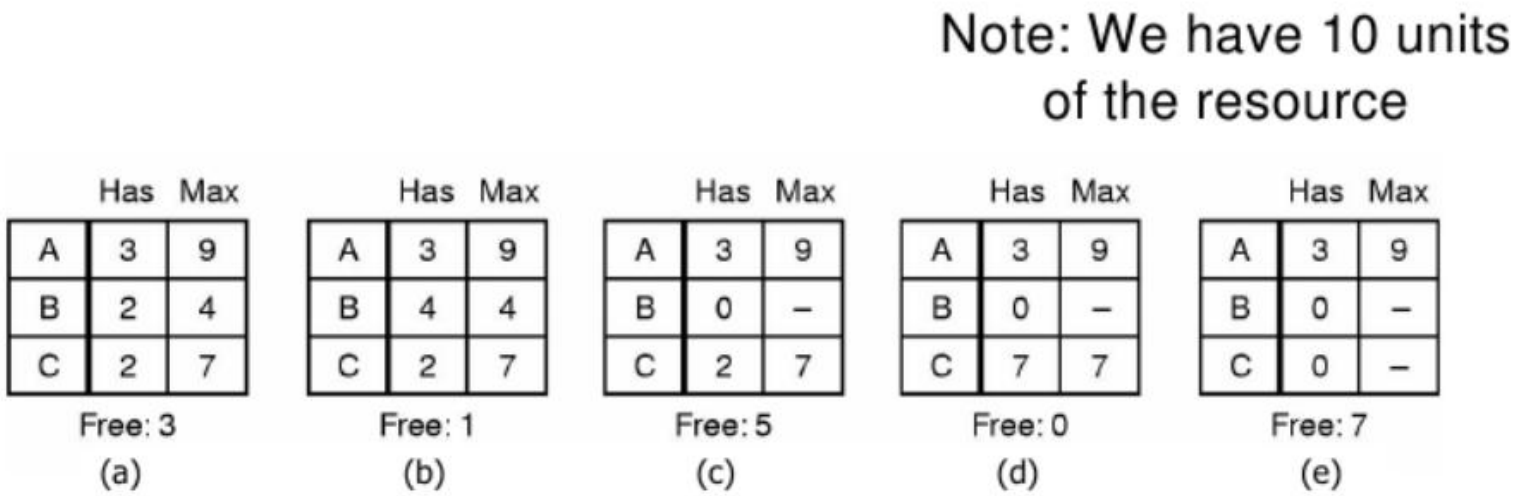


Figure 6: Demonstration that the system state in (a) is safe.

Process-A requests one extra unit resulting in (b)

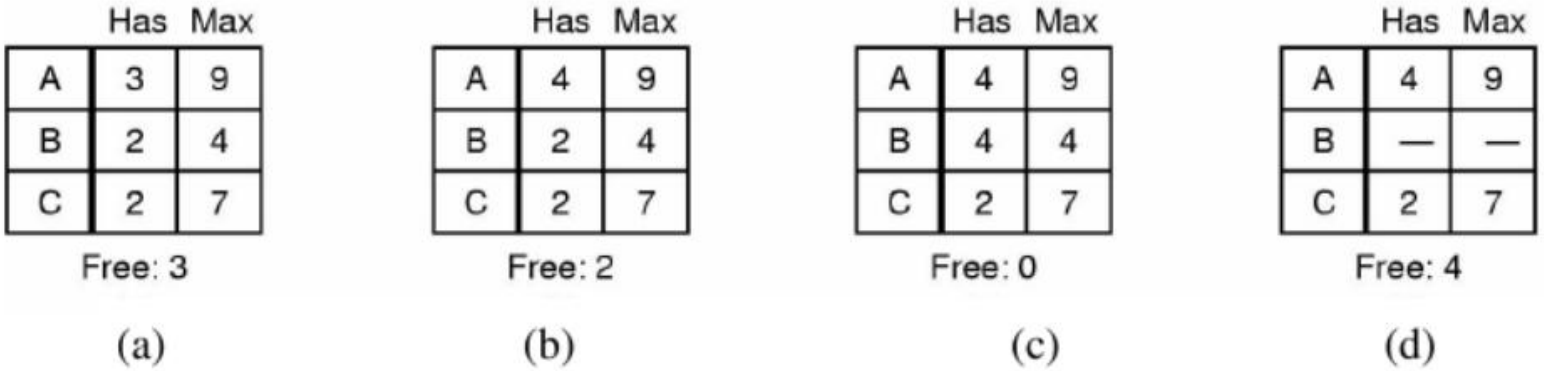


Figure 7: Demonstration that the system state in (b) is unsafe.

Safe and Unsafe States

- Unsafe states are not necessarily deadlocked
  - With a lucky sequence, all process may complete
  - However, we cannot guarantee that they will complete (not deadlock)
- Safe states guarantee we will eventually complete all processes
- Deadlock avoidance algorithm
  - Only grant requests that result in safe states

Bankers Algorithm

- Modeled on a Banker with Customers
  - The banker has a limited amount of money to loan customers
    - Limited number of resources
  - Each customer can borrow money up to the customer's credit limit
    - Maximum number of resources required
- Basic Idea
  - Keep the bank in a safe state
    - So all customers are happy even if they all request to borrow up to their credit limit at the same time.
  - Customers wishing to borrow such that the bank would enter an unsafe state must wait until somebody else repays their loan such that the the transaction becomes safe.

The Banker's Algorithm for a Single Resource

	Has	Max
A	0	6
B	0	5
C	0	4
D	0	7

Free: 10

(a)

	Has	Max
A	1	6
B	1	5
C	2	4
D	4	7

Free: 2

(b)

	Has	Max
A	1	6
B	2	5
C	2	4
D	4	7

Free: 1

(c)

- Three resource allocation state
  - Safe (a)
  - Safe (b)
  - Unsafe (c)

Banker's Algorithm for Multiple Resources

Process	Tape drives	Plotters	Scanners	CD ROMs
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Scanners	CD ROMs
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

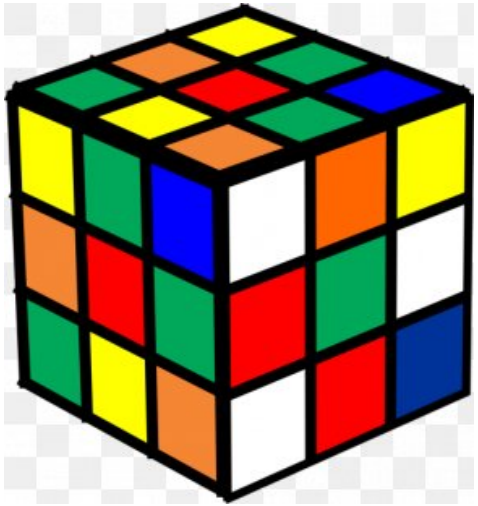
Resources still needed

E = (6342)
P = (5322)
A = (1020)

Figure 8: Example of banker's algorithm with multiple resources

Bankers Algorithm is used rarely in practice

- **It is difficult (sometime impossible) to know in advance**
  - the resources a process will require
  - the number of processes in a dynamic system



Approach 4: Deadlock Prevention

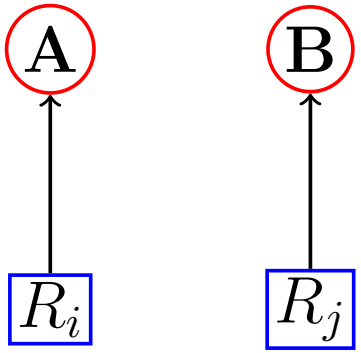
- Attacking the Mutual Exclusion Condition
  - Not feasible
    - Some devices/resource are intrinsically not shareable.
- Attacking the Hold and Wait Condition
  - Require processes to request resources before starting
    - a process never has to wait for what it needs
  - Problems
    - may not know required resources at start of run
    - also ties up resources other processes could be using
  - Variation:
    - process must give up all resources
    - then request all immediately needed

Attacking the No Preemption Condition

- This is not a viable option
- Consider a process given the printer
  - halfway through its job
  - now forcibly take away printer
  - !!??

Attacking the Circular Wait Condition

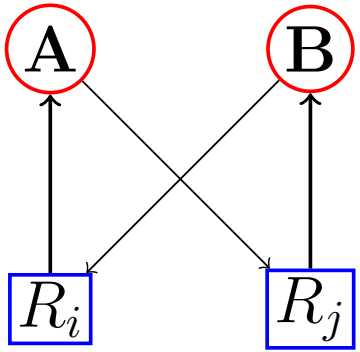
1. Imagesetter
2. Scanner
3. Plotter
4. Tape drive
5. CD-ROM drive



Numerically ordered resources

Attacking the Circular Wait Condition

- The displayed deadlock cannot happen
  - If A requires 1, it must acquire it before acquiring 2
  - Note: If B has 1, all higher numbered resources must be free or held by processes who doesn't need 1



- Resources ordering is a common technique in practice!!!!!

Prevention

Summary of approaches to deadlock prevention

Condition	Approach
Mutual Exclusion	Not feasible
Hold and Wait	Request resource initially
No Preemption	Take resources away
Circular Wait	Order resources

Nonresource Deadlock sand Starvation

Nonresource Deadlocks

- Possible for two processes to deadlock
  - each is waiting for the other to do some task
- Can happen with semaphores
  - semaphores (mutex and another)
  - if done in wrong order, deadlock results

Starvation

- Starvation is where the overall system makes progress, but one or more processes never make progress.
  - Example: An algorithm to allocate a resource may be to give to shortest job first
  - Works great for multiple short jobs in a system
  - May cause long job to be postponed indefinitely, even though not blocked
- Solution:
  - First-come, first-serve policy