Chapter 3

Deadlocks

Resources

- Examples of computer resources
  - printers
  - tape drives
  - tables
- Processes need access to resources in reasonable 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

Resources (1)

- Deadlocks occur when ...
  - 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 (2)

- 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

Introduction to Deadlocks

- Formal definition :

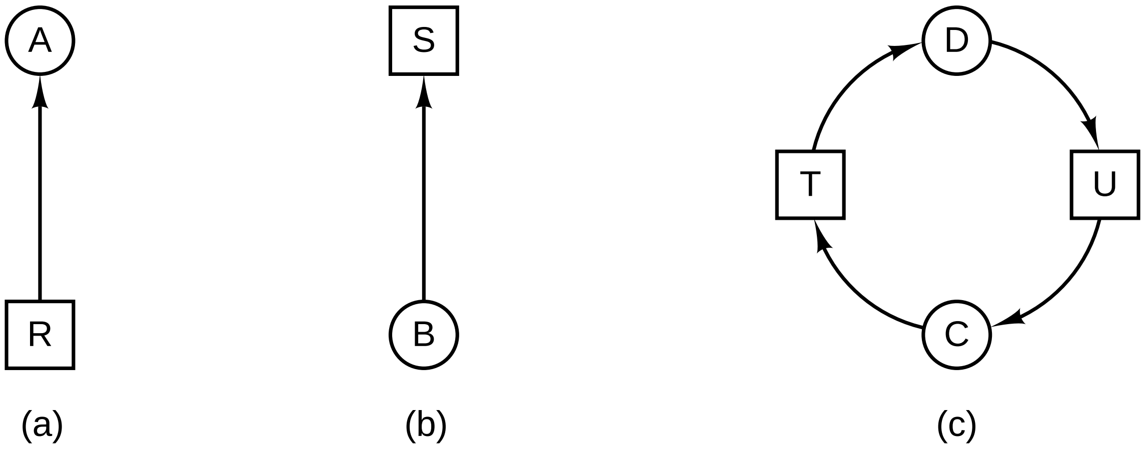
  A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause

- Usually the event is release of a currently held resource
- None of the processes can ...
  - run
  - release resources
  - be awakened

Four Conditions for Deadlock

1. Mutual exclusion condition
    - each resource assigned to 1 process or is available
2. Hold and wait condition
    - process holding resources can request additional
3. No preemption condition
    - previously granted resources cannot forcibly taken away
4. 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 (1)



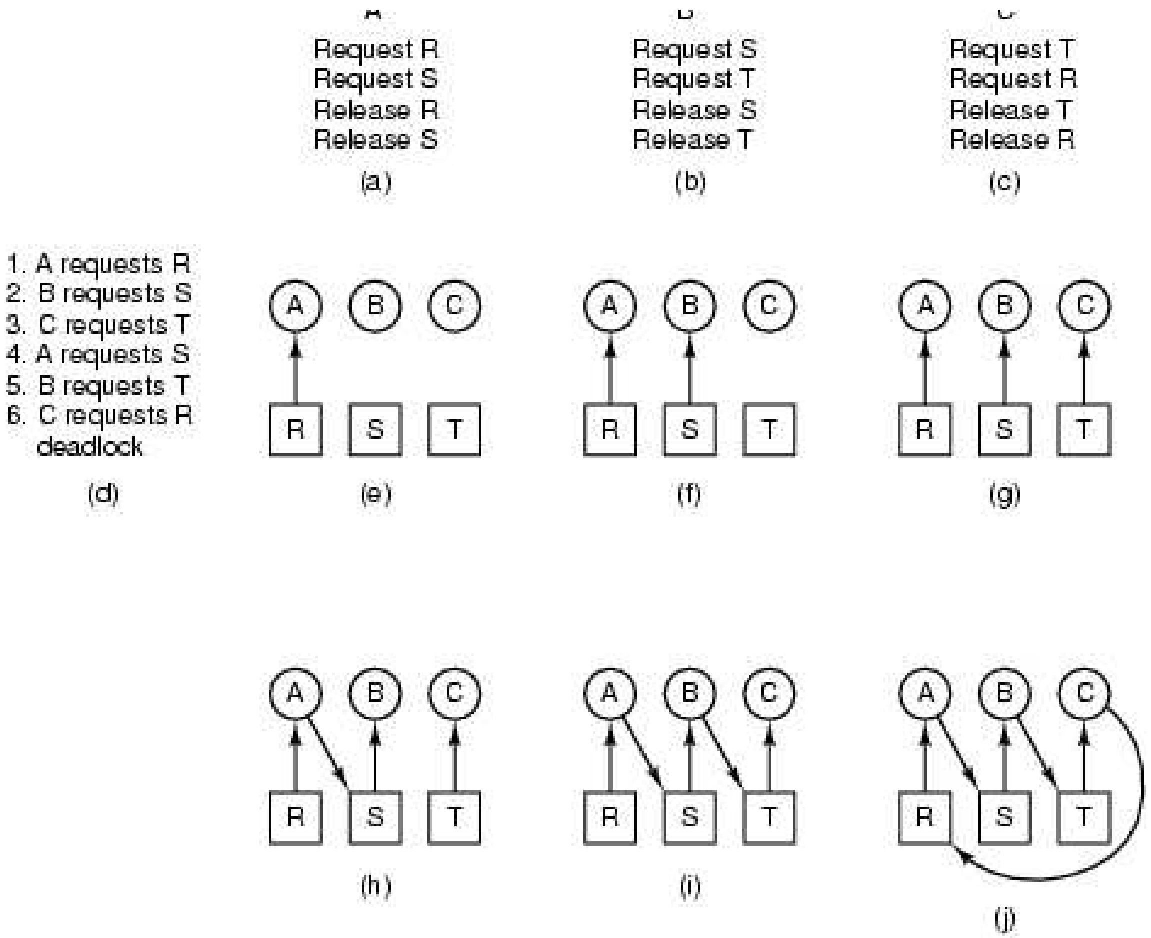
- Modeled with directed graphs
  - (a) resource R assigned to process A
  - (b) process B is requesting/waiting for resource S
  - (c) process C and D are in deadlock over resources T and U

Deadlock Modeling (2)

Strategies for dealing with Deadlocks

1. just ignore the problem altogether
    - Ostrich Algorithm
2. detection and recovery
3. dynamic avoidance
    - careful resource allocation
4. prevention
    - negating one of the four necessary conditions

Deadlock Modeling (3)



Request R
Request S
Release R
Release S

(a)

Request S
Request T
Release S
Release T

(b)

Request T
Request R
Release T
Release R

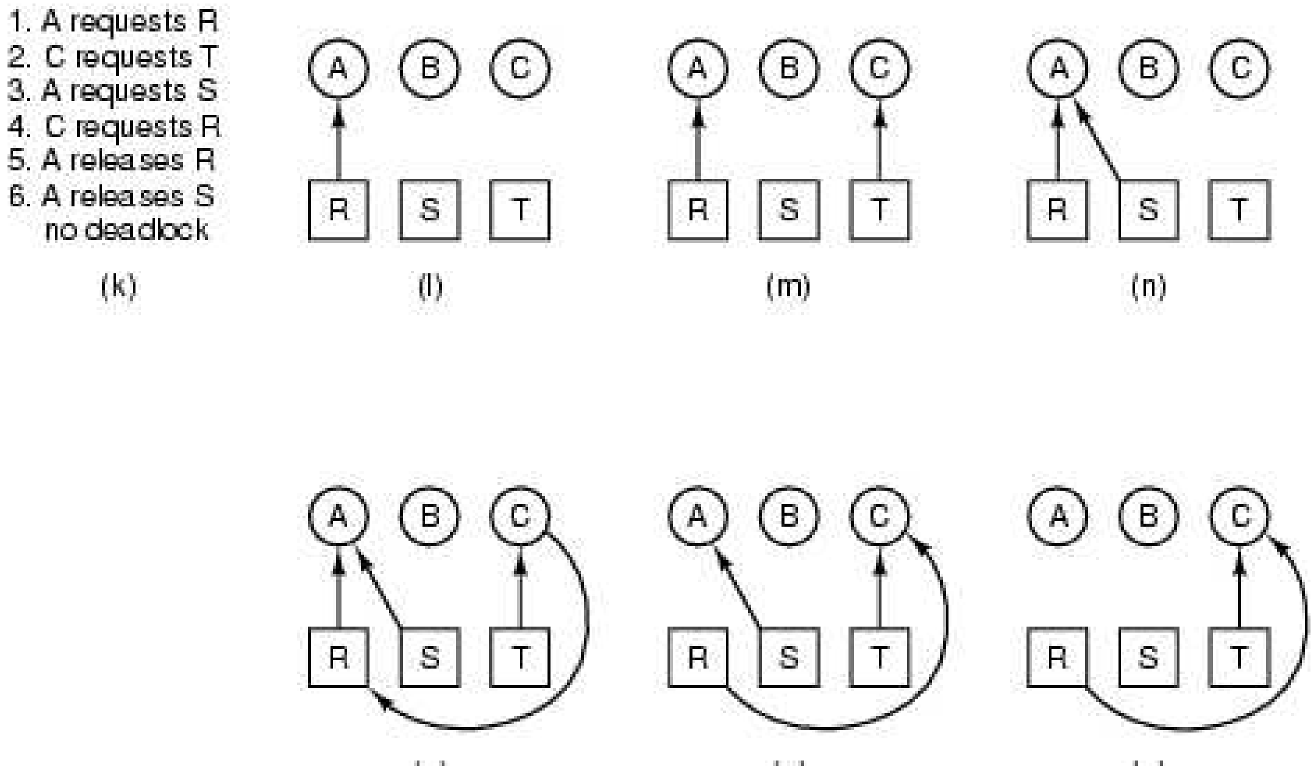
(c)

1. A requests R
2. B requests S
3. C requests T
4. A requests S
5. B requests T
6. C requests R
deadlock

(d)

How deadlock occurs

Deadlock Modeling (4)

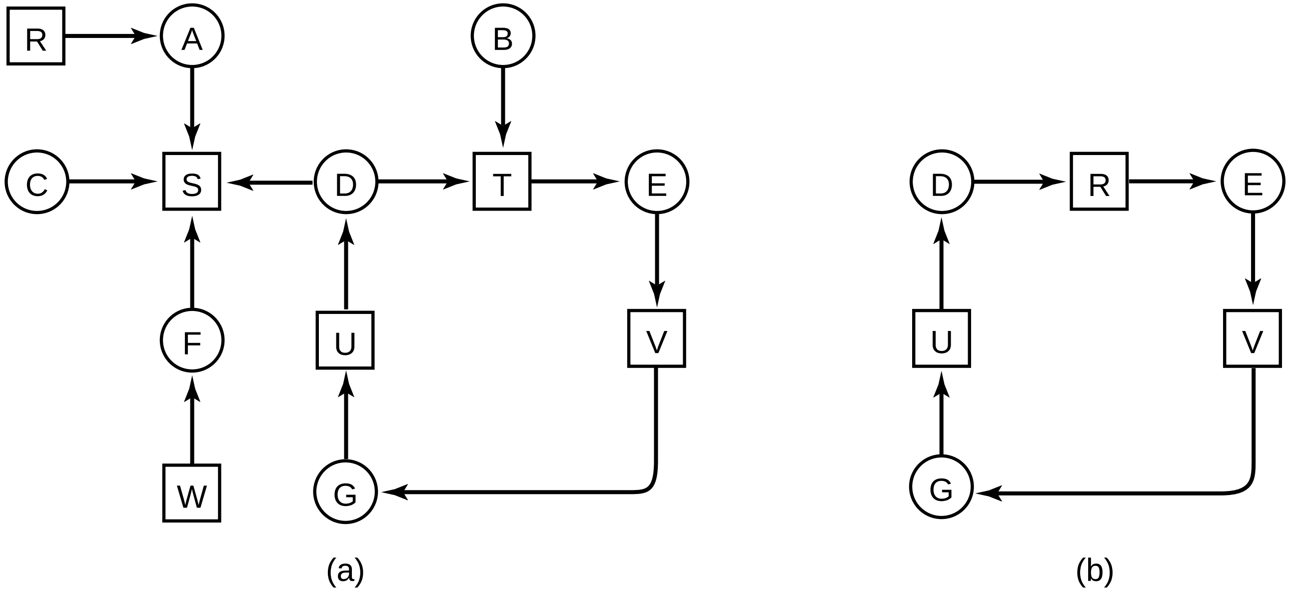


How deadlock can be avoided

The Ostrich Algorithm

- Pretend there is no problem
- Reasonable if
  - deadlocks occur very rarely
  - cost of prevention is high
- UNIX and Windows takes this approach
- It is a trade off between
  - convenience
  - correctness

Detection with One Resource of Each Type (1)



(a) (b)

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

Detection with One Resource of Each Type (2)

Resources in existence
$(E_1, E_2, E_3, \ldots, E_m)$

Resources available
$(A_1, A_2, A_3, \ldots, A_m)$

Current allocation matrix

$$\left.\begin{array}{ccccc} C_{11} & C_{12} & C_{13} & \cdots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nm} \end{array}\right]$$

Row n is current allocation
to process n

Request matrix

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1m} \\ R_{21} & R_{22} & R_{23} & \cdots & R_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{n1} & R_{n2} & R_{n3} & \cdots & R_{nm} \end{bmatrix}$$

Row 2 is what process 2 needs

Data structures needed by deadlock detection algorithm

For vectors A and B (m), $A \leq B$ iff $Ai \leq Bi$ for $1 \leq i \leq m$

Detection with One Resource of Each Type (3)

$$E = \begin{array}{cccc} \text{Tape drives} & \text{Plotters} & \text{Scanners} & \text{CD Roms} \\ (\,4 & 2 & 3 & 1\,) \end{array} \qquad A = \begin{array}{cccc} \text{Tape drives} & \text{Plotters} & \text{Scanners} & \text{CD Roms} \\ (\,2 & 1 & 0 & 0\,) \end{array}$$

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

An example for the deadlock detection algorithm

1. $R2 : (2\ 1\ 0\ 0) \Rightarrow A = (2\ 2\ 2\ 0)$
2. $R1 : (1\ 0\ 1\ 0) \Rightarrow A = (4\ 2\ 2\ 1)$
3. $R0 : (2\ 0\ 0\ 1) \Rightarrow A = (4\ 2\ 3\ 1)$

Recovery from Deadlock (1)

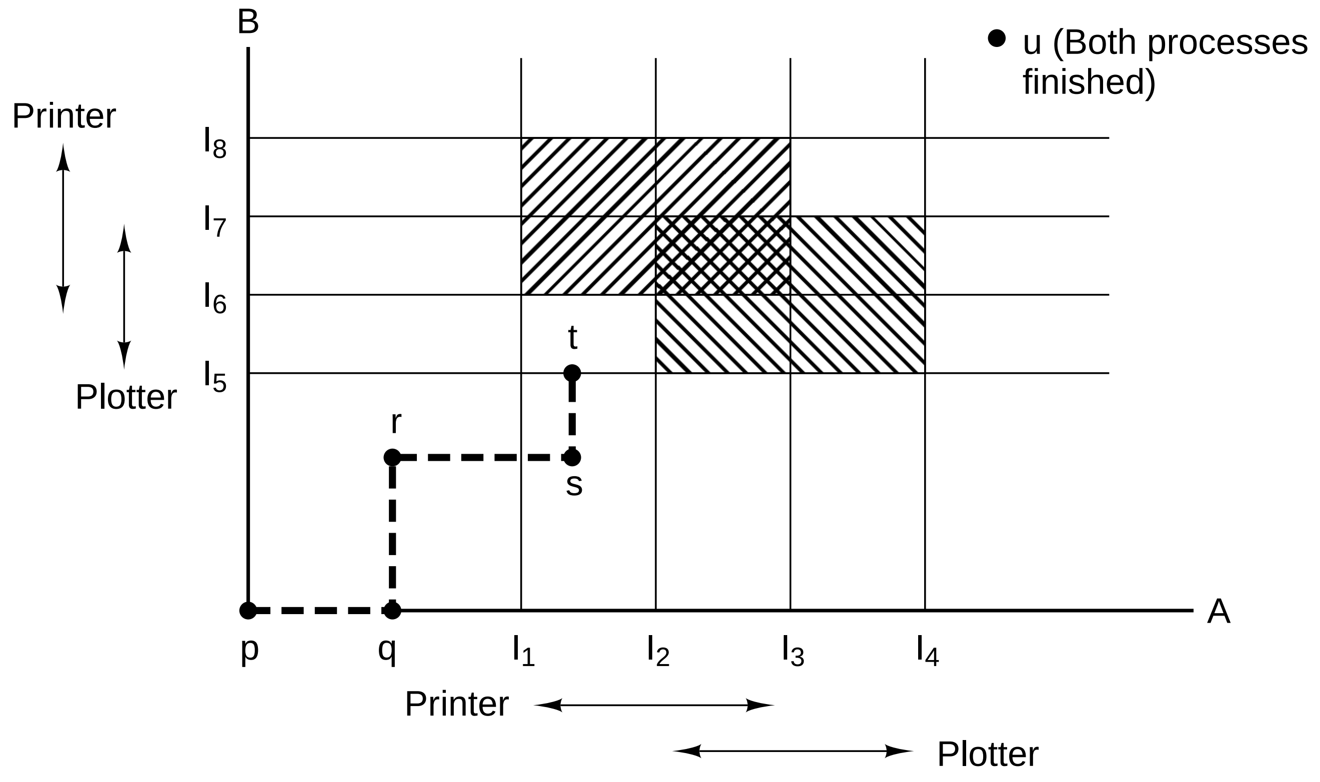
- Recovery through preemption
  - take a resource from some other process
  - depends on nature of the resource
- Recovery through rollback
  - checkpoint a process periodically
  - use this saved state
  - restart the process if it is found deadlocked

Recovery from Deadlock (2)

- 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

Deadlock Avoidance

Resource Trajectories



Two process resource trajectories

Safe and Unsafe States (1)

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3
(a)

	Has	Max
A	3	9
B	4	4
C	2	7

Free: 1
(b)

	Has	Max
A	3	9
B	0	–
C	2	7

Free: 5
(c)

	Has	Max
A	3	9
B	0	–
C	7	7

Free: 0
(d)

	Has	Max
A	3	9
B	0	–
C	0	–

Free: 7
(e)

(b) finished, (c) finished, A can be finished

Demonstration that the state in (a) is safe

1. Not deadlocked
2. ∃ scheduling over each process can request MAX

Safe and Unsafe States (2)

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3
(a)

	Has	Max
A	4	9
B	2	4
C	2	7

Free: 2
(b)

	Has	Max
A	4	9
B	4	4
C	2	7

Free: 0
(c)

	Has	Max
A	4	9
B	—	—
C	2	7

Free: 4
(d)

Demonstration that the sate in b is not safe

deadlock $\subseteq$ unsafe

(a) Give A one more

(d) A: 5 needed, C: 5 needed, only 4 available

The Banker's Algorithm for a Single Resource

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

Free: 10

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

Free: 2

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

Free: 1

- Three resource allocation states

  (a) safe

  – Any order

  (b) safe

  – C(4), B(5), D(9), one possibility

  (c) unsafe

  – None can request MAX

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)

Example of banker's algorithm with multiple resources

- E = Exiting
- P = Possessed
- A = Available

Deadlock Prevention

Attacking the Mutual Exclusion Condition

- Some devices (such as printer) can be spooled
  - only the printer daemon uses printer resource
  - thus deadlock for printer eliminated
- Not all devices can be spooled
  (e.g. process table)
- Principle:
  - avoid assigning resource when not absolutely necessary
  - as few processes as possible actually claim the resource

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)

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

Left: Normally ordered resources

Right: A resource graph

Summary of approaches to deadlock prevention

Condition	**Approach**
Mutual exclusion	Spool everything
Hold and wait	Request all resources initially
No preemption	Take resources away
Circular wait	Order resources numerically

Other Issues

Two-Phase Locking

- Phase One
  - process tries to lock all records it needs, one at a time
  - if needed record found locked, start over
  - (no real work done in phase one)
- If phase one succeeds, it starts second phase,
  - performing updates
  - releasing locks
- Note similarity to requesting all resources at once
- Algorithm works where programmer can arrange
  - program can be stopped, restarted (in this way)

Nonresource Deadlocks

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

Starvation

- Algorithm to allocate a resource
  - may be to give to shortest job first (SJF is scheduling)
- 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
  - can increase priority by wait time