

# Chapter 6

## Deadlocks

- 6.1. Resources
- 6.2. Introduction to deadlocks
- 6.3. The ostrich algorithm
- 6.6. Deadlock prevention
- 6.4. Deadlock detection and recovery
- 6.5. Deadlock avoidance
- 6.7. Other issues

1

## Learning Outcomes

- Understand what deadlock is and how it can occur when giving mutually exclusive access to multiple resources.
- Understand several approaches to mitigating the issue of deadlock in operating systems.
  - Including *deadlock prevention, detection and recovery, and deadlock avoidance.*

2

## Resources

- Examples of computer resources
  - printers
  - tape drives
  - Tables in a database
- Processes need access to resources in reasonable order
- Preemptable resources
  - can be taken away from a process with no ill effects
- Nonpreemptable resources
  - will cause the process to fail if taken away

3

## Resources & Deadlocks

- 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 - *Deadlocked*
- Deadlocks occur when ...
  - processes are granted exclusive access to devices, **locks**, tables, etc..
  - we refer to these entities generally as resources

4

## Resource Access

- 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

5

## Two example resource usage patterns

```
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);
}
```

6

## Introduction to Deadlocks

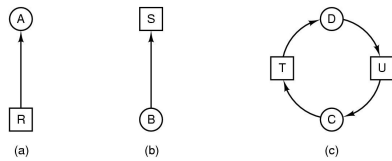
- **Formal definition :**  
*A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause*
- Usually the event is release of a currently held resource
- None of the processes can ...
  - run
  - release resources
  - be awakened

## 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 be 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

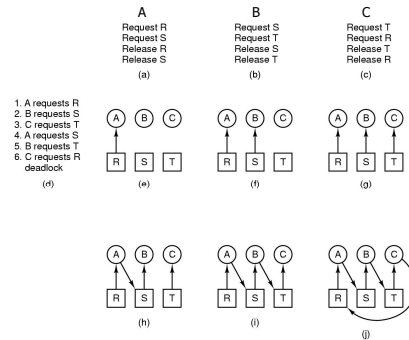
## Deadlock Modeling

- Modeled with directed graphs



- 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

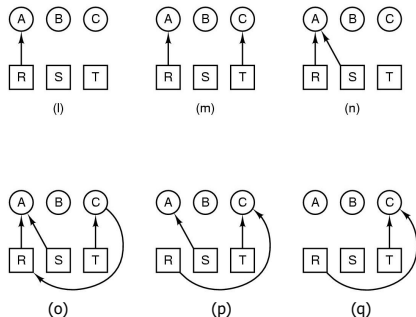
## Deadlock Modeling



How deadlock occurs

## Deadlock Modeling

1. A requests R
  2. C requests T
  3. A requests S
  4. C requests R
  5. A releases R
  6. A releases S
- no deadlock



How deadlock can be avoided

## Deadlock

### Strategies for dealing with Deadlocks

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

## Approach 1: The Ostrich Algorithm

- Pretend there is no problem
- Reasonable if
  - deadlocks occur very rarely
  - cost of prevention is high
    - Example of "cost", only one process runs at a time
- UNIX and Windows takes this approach for some of the more complex resource relationships they manage
- It's a trade off between
  - Convenience (engineering approach)
  - Correctness (mathematical approach)

13 UNSW

13

## Approach 2: Deadlock Prevention

- Resource allocation rules prevent deadlock by prevent one of the four conditions required for deadlock from occurring
  - Mutual exclusion
  - Hold and wait
  - No preemption
  - Circular Wait

14 UNSW

14

## Approach 2 Deadlock Prevention Attacking the Mutual Exclusion Condition

- Not feasible in general
  - Some devices/resource are intrinsically not shareable.

15 UNSW

15

## Attacking the Hold and Wait Condition

- Require processes to request resources before starting
  - a process never has to wait for what it needs
- Issues
  - may not know required resources at start of run
    - => not always possible
  - also ties up resources other processes could be using
- Variations:
  - process must give up all resources if it would block holding a resource
  - then request all immediately needed
  - prone to livelock

16 UNSW

16

## Livelock

- Livelocked processes are not blocked, change state regularly, but never make progress.
- Example: Two people passing each other in a corridor that attempt to step out of each other's way in the same direction, indefinitely.
  - Both are actively changing state
  - Both never pass each other.

17 UNSW

17

## Deadlock example

```
void proc_A() {
    lock_acquire(&res_1);
    lock_acquire(&res_2);
    use_both_res();
    lock_release(&res_2);
    lock_release(&res_1);
}

void proc_B() {
    lock_acquire(&res_2);
    lock_acquire(&res_1);
    use_both_res();
    lock_release(&res_1);
    lock_release(&res_2);
}
```

18 UNSW

18

## Livelock example

```

void proc_A() {
    lock_acquire(&res_1);
    while(try_lock(&res_2) == FAIL) {
        lock_release(&res_1);
        wait_fixed_time();
        lock_acquire(&res_1);
    }
    use_both_res();
    lock_release(&res_2);
    lock_release(&res_1);
}

void proc_B() {
    lock_acquire(&res_2);
    while(try_lock(&res_1) == FAIL) {
        lock_release(&res_2);
        wait_fixed_time();
        lock_acquire(&res_2);
    }
    use_both_res();
    lock_release(&res_1);
    lock_release(&res_2);
}

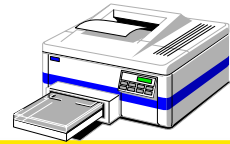
```

19 UNSW

19

## 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
  - !!??



20 UNSW

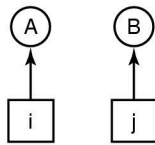
20

## Attacking the Circular Wait Condition

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

(a)

- Numerically ordered resources



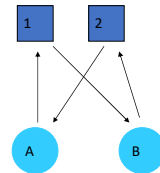
(b)

21 UNSW

21

## Attacking the Circular Wait Condition

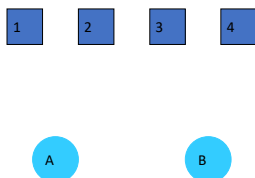
- The displayed deadlock cannot happen
  - If A requires 1, it must acquire it before acquiring 2
  - Note: If B has 1, all higher numbered resources must be free or held by processes who doesn't need 1
- Resources ordering is a common technique in practice!!!!



22 UNSW

22

## Example



23 UNSW

23

## Summary of approaches to deadlock prevention

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

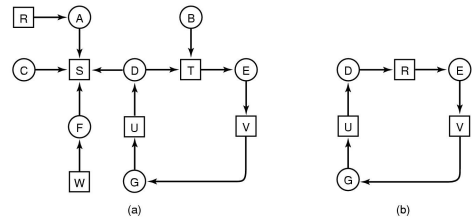
24 UNSW

24

### Approach 3: Detection and Recovery

- Need a method to determine if a system is deadlocked.
- Assuming deadlocked is detected, we need a method of recovery to restore progress to the system.

### Approach 3 Detection with One Resource of Each Type

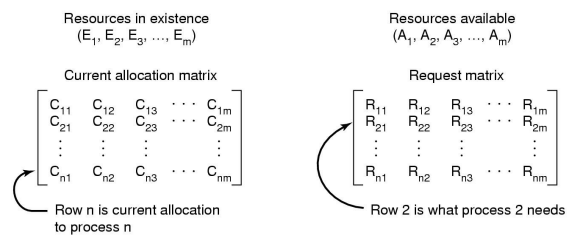


- Note the resource ownership and requests
- A cycle can be found within the graph, denoting deadlock

### What about resources with multiple units?

- Some examples of multi-unit resources
  - RAM
  - Blocks on a hard disk drive
  - Slots in a buffer
- We need an approach for dealing with resources that consist of more than a single unit.

### Detection with Multiple Resources of Each Type



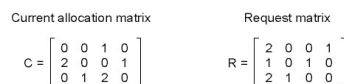
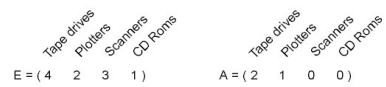
Data structures needed by deadlock detection algorithm

### Note the following invariant

Sum of current resource allocation + resources available = resources that exist

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

### Detection with Multiple Resources of Each Type



An example for the deadlock detection algorithm

### Detection Algorithm

1. Look for an unmarked process  $P_i$ , 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  $P_i$ . Go to step 1
  3. If no such process exists, terminate.
- Remaining processes are deadlocked

### Example Deadlock Detection

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

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

### Example Deadlock Detection

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

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

### Example Deadlock Detection

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

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

### Example Deadlock Detection

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

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

### Example Deadlock Detection

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

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

### Example Deadlock Detection

$$E = (4 \ 2 \ 3 \ 1)$$

$$A = (4 \ 2 \ 2 \ 1)$$

$$\begin{matrix} \rightarrow \\ \rightarrow \\ \rightarrow \end{matrix} C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix}$$

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

37 UNSW

37

### Example Deadlock Detection

$$E = (4 \ 2 \ 3 \ 1)$$

$$A = (4 \ 2 \ 2 \ 1)$$

$$\begin{matrix} \rightarrow \\ \rightarrow \\ \rightarrow \end{matrix} C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix}$$

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

38 UNSW

38

### Example Deadlock Detection

$$E = (4 \ 2 \ 3 \ 1)$$

$$A = (4 \ 2 \ 3 \ 1)$$

$$\begin{matrix} \rightarrow \\ \rightarrow \\ \rightarrow \end{matrix} C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{pmatrix}$$

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

39 UNSW

39

### Example Deadlock Detection

- Algorithm terminates with no unmarked processes
- We have no dead lock

40 UNSW

40

### Example 2: Deadlock Detection

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

$$E = (4 \ 2 \ 3 \ 1)$$

$$A = (2 \ 1 \ 0 \ 0)$$

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

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

41 UNSW

41

### 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
    - No guarantee is won't deadlock again

42 UNSW

42

## Recovery from Deadlock

- 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

43 UNSW

43

## Approach 4

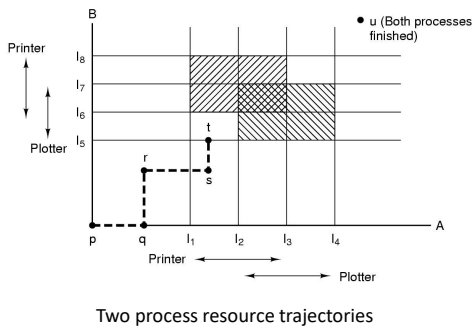
### Deadlock Avoidance

- 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

44 UNSW

44

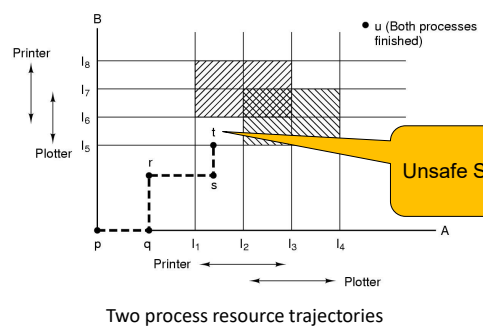
## Deadlock Avoidance Resource Trajectories



45 UNSW

45

## Deadlock Avoidance Resource Trajectories



46 UNSW

46

## Safe and Unsafe States

- A 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*

47 UNSW

47

## Safe and Unsafe States

Note: We have 10 units of the resource

	Has	Max		Has	Max		Has	Max		Has	Max		Has	Max
A	3	9	A	3	9	A	3	9	A	3	9	A	3	9
B	2	4	B	4	4	B	0	-	B	0	-	B	0	-
C	2	7	C	2	7	C	2	7	C	7	7	C	0	-
	Free: 3			Free: 1			Free: 5			Free: 0			Free: 7	
	(a)			(b)			(c)			(d)			(e)	

Demonstration that the state in (a) is safe

48 UNSW

48



## Safe and Unsafe States

A requests one extra unit resulting in (b)

	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 state in b is not safe

## Safe and Unsafe State

- Unsafe states are not necessarily deadlocked
  - With a lucky sequence, all processes 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

- Modelled 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 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 states

- safe
- safe
- unsafe

B requests one more, should we grant it?

## 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
- Problem is structured similar to deadlock detection with multiple resources.
- Example in tutorial

## Bankers Algorithm is not commonly used in practice

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

## Starvation

- A process never receives the resource it is waiting for, despite the resource (repeatedly) becoming free, the resource is always allocated to another waiting process.
  - Example: An algorithm to allocate a resource may be to give the resource to the shortest job first
  - Works great for multiple short jobs in a system
  - May cause a long job to wait indefinitely, even though not blocked.
- One solution:
  - First-come, first-serve policy