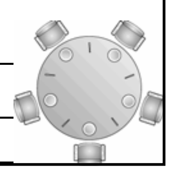
Concurrency

- Reading: OS Concepts pp.243-266
- Contents
  - Introduction
  - Deadlock Prevention
  - Deadlock Avoidance
  - Deadlock Detection & Recovery

The problem

- We only have concurrency problems if
  - ______________________________
  - ______________________________
    ____
- The problems are made worse
  - ______________________________
    ____
  - ______________________________
- The main problems are
  - ______________________________
  - ______________________________
    ____



Introduction
Prevention
Avoidance
Detection

How OSes deal with it.

- Resource Classes
  - ______________________________
  - ______________________________
- Possible restictions (to simplify the problem):
  - ______________________________
  - ______________________________
- Facilities provided for users & kernel
- Solutions
  - ______________________________
    ____
  - ______________________________

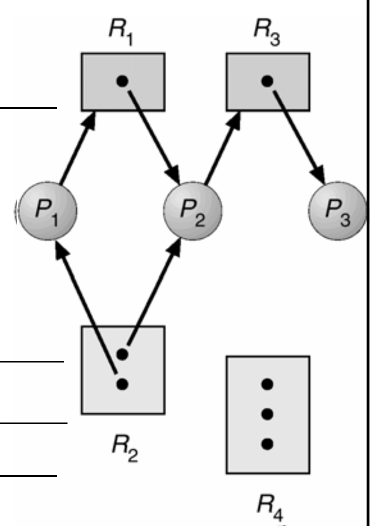
Introduction
Prevention
Avoidance
Detection

Resource Allocation Graphs.

- A way of presenting resource usage issues
- Made up of

1. ____________________
   - $P = \{P_1, P_2, \ldots, P_n\}$
   - $R = \{R_1, R_2, \ldots, R_m\}$
2. ____________________
   - $P_i \rightarrow R_j$ ____________
   - $R_j \rightarrow P_i$ ____________

- ____________________

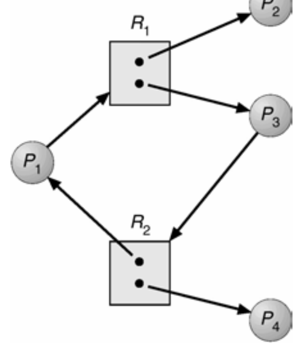


Process Management, Topic 5, Concurrency Problems

Introduction
Prevention
Avoidance
Detection

When is there deadlock?

- Deadlock results from cycle in the resource alllocation graphs
- No Cycles ⇒ ______________
- A cycle ⇒
  - ________________________
    => __________________
  - ________________________
    => __________________



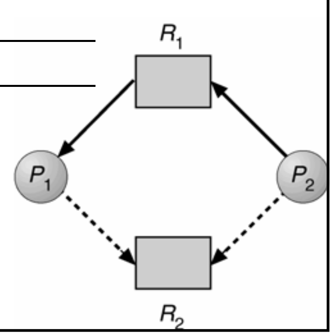
5

Process Management, Topic 5, Concurrency Problems

Introduction
Prevention
Avoidance
Detection

Possibility of Deadlock

- Extension for deadlock avoidance…
- ____
- ____
- ____
  ____
- ______________________________
  ____
- ________________________
  ____
  ____



6

Deadlock Characteristics.

- 4 conditions required for deadlock:
  1. Mutual exclusion: ________________________
     ____
  2. Hold and wait: ____________________________
     ____
     ____
  3. No preemption: ___________________________
     ____
     ____
  4. Circular wait: ____________________________
     - $P_0 \rightarrow R_a \rightarrow P_1 \rightarrow R_b \rightarrow \ldots \rightarrow P_n \rightarrow R_z \rightarrow P_0$

Deadlock Prevention

- Prevent one of the four conditions
  1. Mutual exclusion
     - ______________________________________
     - ______________________________________
  2. Hold and wait
     - ______________________________________
       ____
       ____
       - ____________________________________
       - ____________________________________
     - ______________________________________
       ____

Deadlock Prevention (contd)

Introduction
Prevention
Avoidance
Detection

3. No preemption
    - ______________________________
      ____
    - ______________________________
      ____
    - ______________________________
4. Circular wait
    - ______________________________
    - ______________________________

Process Management, Topic 5, Concurrency Problems

9

Deadlock Prevention (contd)

Introduction
Prevention
Avoidance
Detection

- Deadlock prevention problems…
    - ______________________________
    - ______________________________

Process Management, Topic 5, Concurrency Problems

10

Deadlock Avoidance.

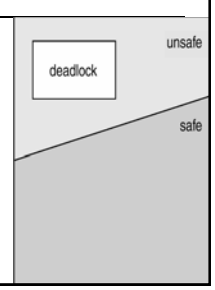
- Attempts to be less restrictive than deadlock prevention

Process Management, Topic 5, Concurrency Problems

Introduction
Prevention
Avoidance
Detection

Safe States

- Deadlock avoidance always keeps the system in a **safe state**
  - $<P_1, P_2, \ldots, P_n>$



Process Management, Topic 5, Concurrency Problems

Introduction
Prevention
Avoidance
Detection

Banker's Algorithm Theory

- ______________________
- ______________________
  ____
- Banker's algorithm evaluates requests by
  - ______________________
  - ______________________
  - If safe
    - ______________________
    - ______________________

13

Process Management, Topic 5, Concurrency Problems

Introduction
Prevention
Avoidance
Detection

Banker's Safety Algorithm

Using Total[m], Available[m]

Using Max[n][m], Need[n][m], Allocation[n][m]

Define Work[m] and Finish[n]

1. Initialize Work[m] and Finish[n]
2. Find any i such that
   (a) Finish [i] = false
   (b) Need[i] $\leq$ Work
   If no such i exists, go to step 4.
3. Work = Work + Allocation[i]
   Finish[i] = true
   go to step 2.
4. If Finish [i] = true for all i
   then …
   else …

14

Testing Safe State Example

- $P_0 \ldots P_4$
- A * 10, B * 5, C * 7
- *State at T_0:*

	Allocation	*Max*
	A B C	A B C
P_0	0 1 0	7 5 3
P_1	2 0 0	3 2 2
P_2	3 0 2	9 0 2
P_3	2 1 1	2 2 2
P_4	0 0 2	4 3 3

Banker's Algorithm Example

- State at T_0:

	Allocation	*Max*	*Need*	*Available*
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	7 4 3	3 3 2
P_1	2 0 0	3 2 2	1 2 2	
P_2	3 0 1	9 0 5	6 0 2	
P_3	2 1 1	2 2 2	0 1 1	
P_4	0 0 2	4 3 3	4 3 1	

- Request:
  - T_1: P_2 requests (2, 0, 1)
  - T_2: P_4 requests (1, 3, 0)
  - T_3: P_0 requests (0, 2, 0)

Process Management, Topic 5, Concurrency Problems

Introduction
Prevention
Avoidance
Detection

Detection & Recovery

- Allow system to become deadlocked
- ______________________________________
  ____
  ____
- Recovery:
  - ____________________________________
  - ____________________________________
- ______________________________________
  ____
- ______________________________________
  ____

17

Process Management, Topic 5, Concurrency Problems

Introduction
Prevention
Avoidance
Detection

Detection Algorithm

Using Total[m], Available[m]

Using Max[n][m], Need[n][m], Allocation[n][m], Request[n][m

Define Work[m] and Finish[n]

1. Initialize Work[m] and Finish[n]
2. Find any i such that
   (a) Finish [i] = false
   (b) Request[i] ≤ Work
   If no such i exists, go to step 4.
3. Work = Work + Allocation[i]
   Finish[i] = true
   go to step 2.
4. If Finish [i] = true for all i
   then …
   else …

18

Process Management, Topic 5, Concurrency Problems

19

Testing Detection Example

Introduction
Prevention
Avoidance
Detection

- $P_0 \ldots P_4$
- A * 7, B * 2, C * 6
- *State at T_0:*

	Allocation	Request
	A B C	A B C
P_0	0 1 0	0 0 0
P_1	2 0 0	2 0 1
P_2	1 0 3	0 0 2
P_3	2 1 1	1 0 0
P_4	0 0 2	0 0 2

Process Management, Topic 5, Concurrency Problems

20

Summary

- The problem
- Resource Allocation Graphs
- Preventing Deadlock
- Avoiding dealock
- Detecting deadlock