Capítulo 3

Deadlocks - Impasses

3.1. Recurso
3.2. Introdução aos *deadlocks*
3.3. Algoritmo do avestruz
3.4. Detecção e recuperação de *deadlocks*
3.5. Evitando *deadlocks*
3.6. Prevenção de *deadlocks*

Recursos

- Exemplos de recursos de computador
  - impressoras
  - unidades de fita
  - tabelas
- Processos precisam de acesso aos recursos numa ordem racional
- Suponha que um processo detenha o recurso A e solicite o recurso B
  - ao mesmo tempo um outro processo detém B e solicita A
  - ambos são bloqueados e assim permanecem

Recursos

Recursos (1)

- *Deadlocks* ocorrem quando ...
  - garante-se aos processos acesso exclusivo aos dispositivos
  - esses dispositivos são normalmente chamados de recursos
- Recursos preemptíveis
  - podem ser retirados de um processo sem quaisquer efeitos prejudiciais
- Recursos não preemptíveis
  - vão induzir o processo a falhar se forem retirados

Recursos (2)

- Seqüência de eventos necessários ao uso de um recurso
  1. solicitar o recurso
  2. usar o recurso
  3. liberar o recurso
- Deve esperar se solicitação é negada
  - processo solicitante pode ser bloqueado
  - pode falhar resultando em um código de erro

Introdução aos *Deadlocks*

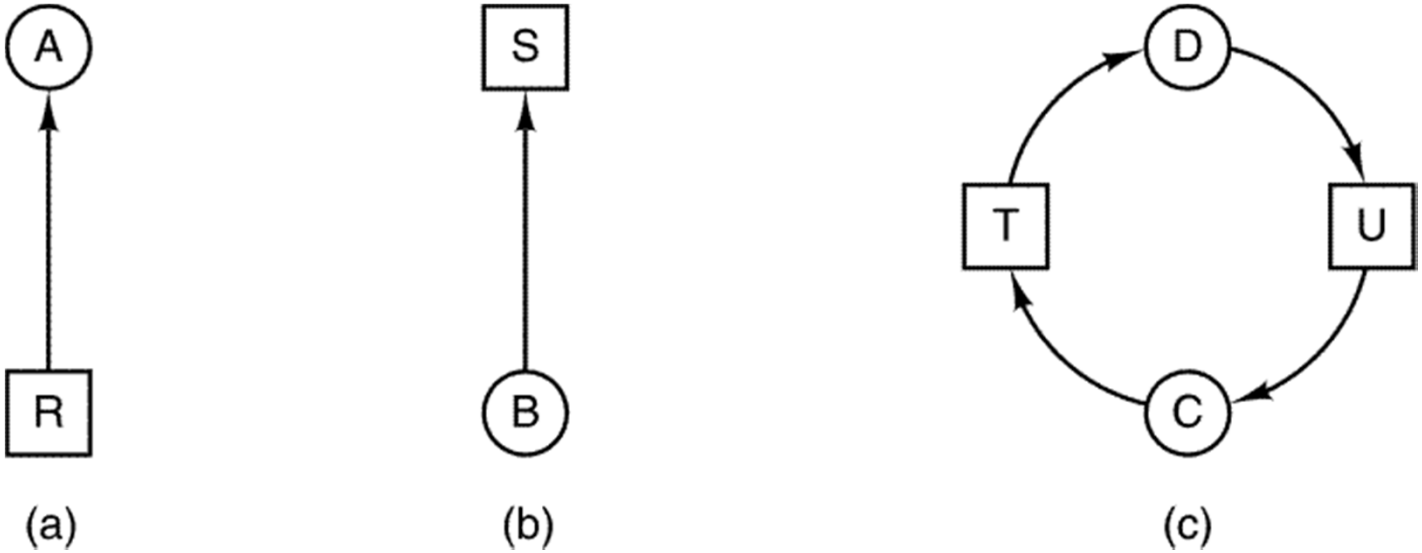
- Definição formal:
  Um conjunto de processos está em situação de deadlock se todo processo pertencente ao conjunto estiver esperando por um evento que somente um outro processo desse mesmo conjunto poderá fazer acontecer
- Normalmente o evento é a liberação de um recurso atualmente retido
- Nenhum dos processos pode...
  - executar
  - liberar recursos
  - ser acordado

Quatro Condições para *Deadlock*

1. Condição de exclusão mútua
   - todo recurso está ou associado a um processo ou disponível
2. Condição de posse e espera
   - processos que retêm recursos podem solicitar novos recursos
3. Condição de não preempção
   - recursos concedidos previamente não podem ser forçosamente tomados
4. Condição de espera circular
   - deve ser uma cadeia circular de 2 ou mais processos
   - cada um está à espera de recurso retido pelo membro seguinte dessa cadeia

Modelagem de *Deadlock* (2)

- Modelado com grafos dirigidos



a) recurso R alocado ao processo A
b) processo B está solicitando/esperando pelo recurso S
c) processos C e D estão em *deadlock* sobre recursos T e U

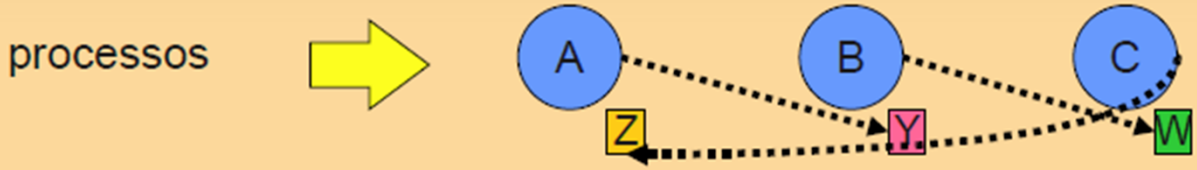
Modelagem de *Deadlock* (3)

Estratégias para tratar *Deadlocks*

1. ignorar por completo o problema
2. detecção e recuperação
3. evitação dinâmica
   - alocação cuidadosa de recursos
4. prevenção
   - negação de uma das quatro condições necessárias

Modelagem de *Deadlock* (4)

- Exemplo:
  - O processo “A” espera pelo processo “B”, que espera pelo processo “C”, que espera pelo processo “A”.



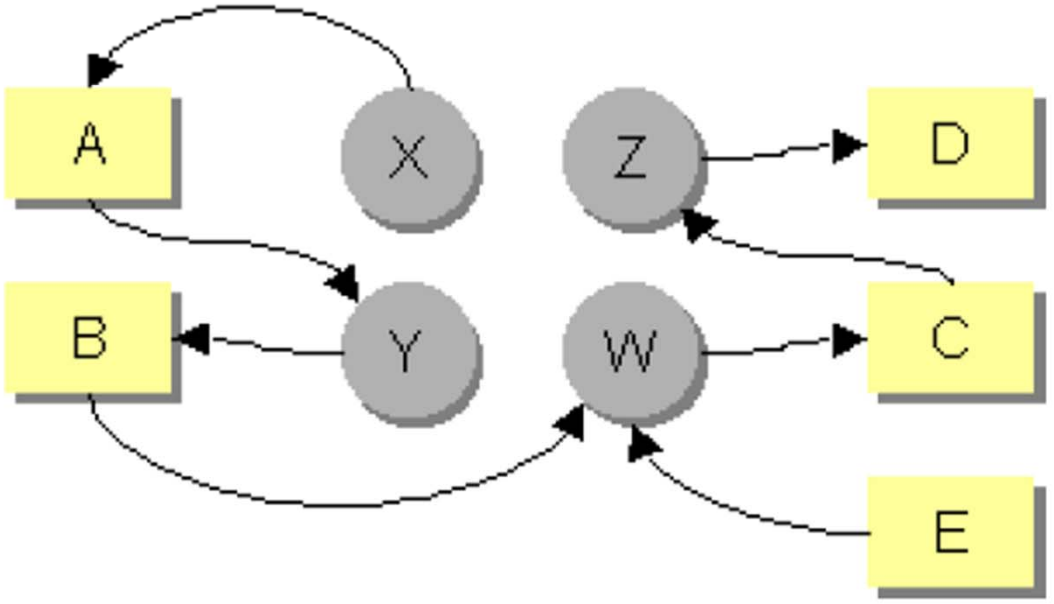
Como ocorre um *deadlock*

Algoritmo do Avestruz

- Finge que o problema não existe
- Razoável se
  - deadlocks ocorrem muito raramente
  - custo da prevenção é alto
- UNIX e Windows seguem esta abordagem
- É uma ponderação entre
  - conveniência
  - correção

Detecção com um Recurso de Cada Tipo (1)

Recurso	Processo Alocado	Processo(s) Requisitante
X	A	
Y	B	A
W	C	B, E
Z	D	C

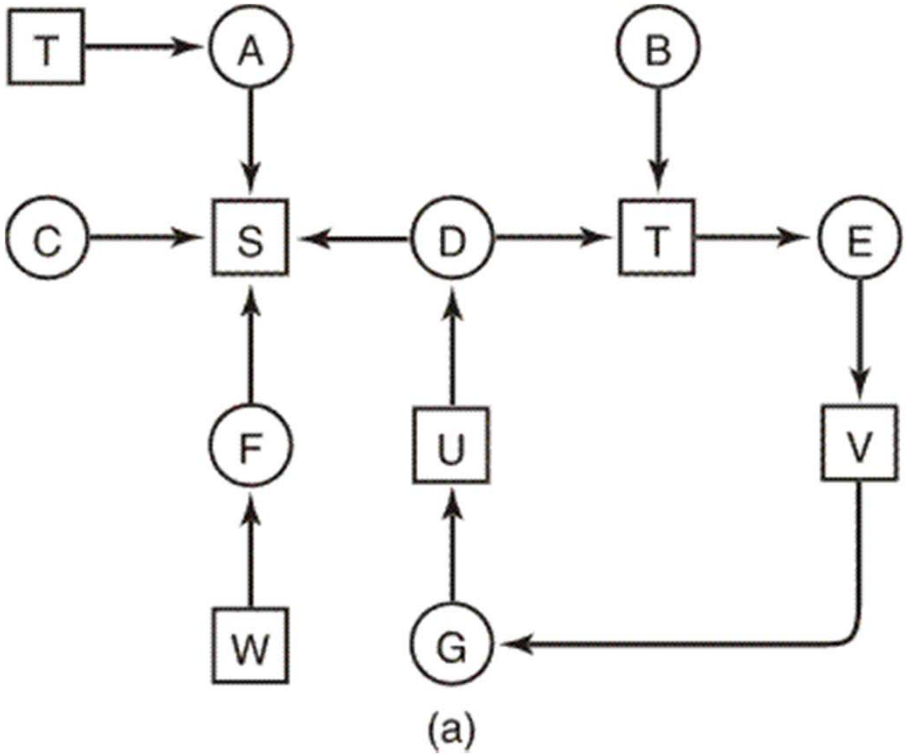


Detecção com um Recurso de Cada Tipo (1)

Neste exemplo, embora exista disputa por recursos (processos B e E requisitam o uso do recurso W), não existe nenhum caminho fechado, ou seja, não existe qualquer *deadlock*, sendo assim este diagrama pode ser reduzido:

1. D finaliza o uso de Z.
2. Com Z livre, C libera W para alocar Z.
3. Com W livre, ou B ou E poderão alocá-lo. Se B for favorecido nesta disputa, alocando W, embora E permaneça em espera, Y será liberado.
4. Com Y livre, A libera X para alocar Y.
5. Não surgindo novos processos, B libera W.
6. Com W livre, E pode prosseguir sua execução.

Detecção com um Recurso de Cada Tipo (1)



(a)

É possível ocorrer um Deadlock ?

- Observe a posse e solicitações de recursos
- Um ciclo pode ser encontrado dentro do grafo, denotando deadlock

Detecção com um Recurso de Cada Tipo (2)

4 unidades de fita;
2 *plotter*;
3 impressoras;
1 unidade de CD-ROM

Três processos:
P_1 usa uma impressora;
P_2 usa duas unidades de fita e uma de CD-ROM;
P_3 usa um *plotter* e duas impressoras;
Cada processo pode precisar de outros recursos (em R);

Recursos existentes
E = (4 2 3 1)
UF P I UCD

Recursos disponíveis
A = (2 1 0 0)
UF P I UCD

Matriz de alocação

	UF	P	I	UCD	
C =	0	0	1	0	P_1
	2	0	0	1	P_2
	0	1	2	0	P_3

Recursos

Matriz de requisições

	UF	P	I	UCD	
R =	0	0	0	0	P_1
	0	0	0	0	P_2
	0	0	0	0	P_3

Estruturas de dados necessárias ao algoritmo de detecção de deadlock

Detecção com um Recurso de Cada Tipo (2)

4 unidades de fita;
2 *plotter*;
3 impressoras;
1 unidade de CD-ROM

Requisições:
P_1 requisita duas unidades de fita e um CD-ROM;
P_2 requisita uma unidade de fita e uma impressora;
P_3 requisita duas unidades de fita e um *plotter*;

Comparamos cada processo em R com os recursos em A, de modo a encontrar um que possa ser rodado.

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (2 1 0 0) **P_3 pode rodar**

Matriz de alocação

$$C = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow \mathbf{P_3} \end{matrix}$$

Estruturas de dados necessárias ao algoritmo de detecção de deadlock

Detecção com um Recurso de Cada Tipo (2)

Comparamos cada processo em R com os recursos em A, de modo a encontrar um que possa ser rodado.

Somente P_3 pode

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (2 1 0 0) **P_3 pode rodar**
A = (**0 0 0 0**)

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ \mathbf{2} & \mathbf{2} & \mathbf{2} & \mathbf{0} \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow \mathbf{P_3} \end{matrix}$$

Estruturas de dados necessárias ao algoritmo de detecção de deadlock

Detecção com um Recurso de Cada Tipo (2)

Ao terminar, P_3 devolve os recursos usados ao vetor de recursos disponíveis.

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (2 1 0 0)
A = (**2 2 2 0**)

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow \mathbf{P_3} \end{matrix}$$

Estruturas de dados necessárias ao algoritmo de detecção de deadlock

Detecção com um Recurso de Cada Tipo (2)

Ao terminar, P_3 devolve os recursos usados ao vetor de recursos disponíveis.

P2 pode então rodar

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (2 1 0 0)
A = (2 2 2 0) **P_2 pode rodar**
A = (**1 2 1 0**)

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ \mathbf{3} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow \mathbf{P_2} \\ \leftarrow P_3 \end{matrix}$$

Estruturas de dados necessárias ao algoritmo de detecção de deadlock

Detecção com um Recurso de Cada Tipo (2)

Ao terminar, P_3 devolve os recursos usados ao vetor de recursos disponíveis.

P_2 pode então rodar

Ao terminar, também ele devolve os recursos

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (2 1 0 0)
A = (2 2 2 0)
A = (**4 2 2 1**)

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow \mathbf{P_2} \\ \leftarrow P_3 \end{matrix}$$

Estruturas de dados necessárias ao algoritmo de detecção de deadlock

Detecção com um Recurso de Cada Tipo (2)

Ao terminar, P_3 devolve os recursos usados ao vetor de recursos disponíveis.

P_2 pode então rodar

Ao terminar, também ele devolve os recursos

Finalmente, o último processo pode rodar – não há deadlock

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (2 1 0 0)
A = (2 2 2 0)
A = (2 2 1 0) **P_1 pode rodar**

Matriz de alocação

$$C = \begin{bmatrix} \mathbf{2} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow \mathbf{P_1} \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Estruturas de dados necessárias ao algoritmo de detecção de deadlock

Detecção com um Recurso de Cada Tipo (2)

Ao final da execução, temos:

4 unidades de fita;
2 *plotters*;
3 impressoras;
1 unidade de CD-ROM

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (4 2 3 1)

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Estruturas de dados necessárias ao algoritmo de detecção de deadlock

Detecção com um Recurso de Cada Tipo (3)

4 unidades de fita;
2 *plotters*;
3 impressoras;
1 unidade de CD-ROM

Requisições:
P_2 requisita duas unidade de fita, uma impressora e uma unidade de CD-ROM;

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (2 1 0 0)

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 2 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow \mathbf{P_2} \\ \leftarrow P_3 \end{matrix}$$

Estruturas de dados necessárias ao algoritmo de detecção de deadlock (2º Caso)

Detecção com um Recurso de Cada Tipo (3)

4 unidades de fita;
2 *plotters*;
3 impressoras;
1 unidade de CD-ROM

Requisições:
P_2 requisita duas unidade de fita, uma impressora e uma unidade de CD-ROM;

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (2 1 0 0) P_3 pode rodar
A = (0 0 0 0)

Matriz de alocação

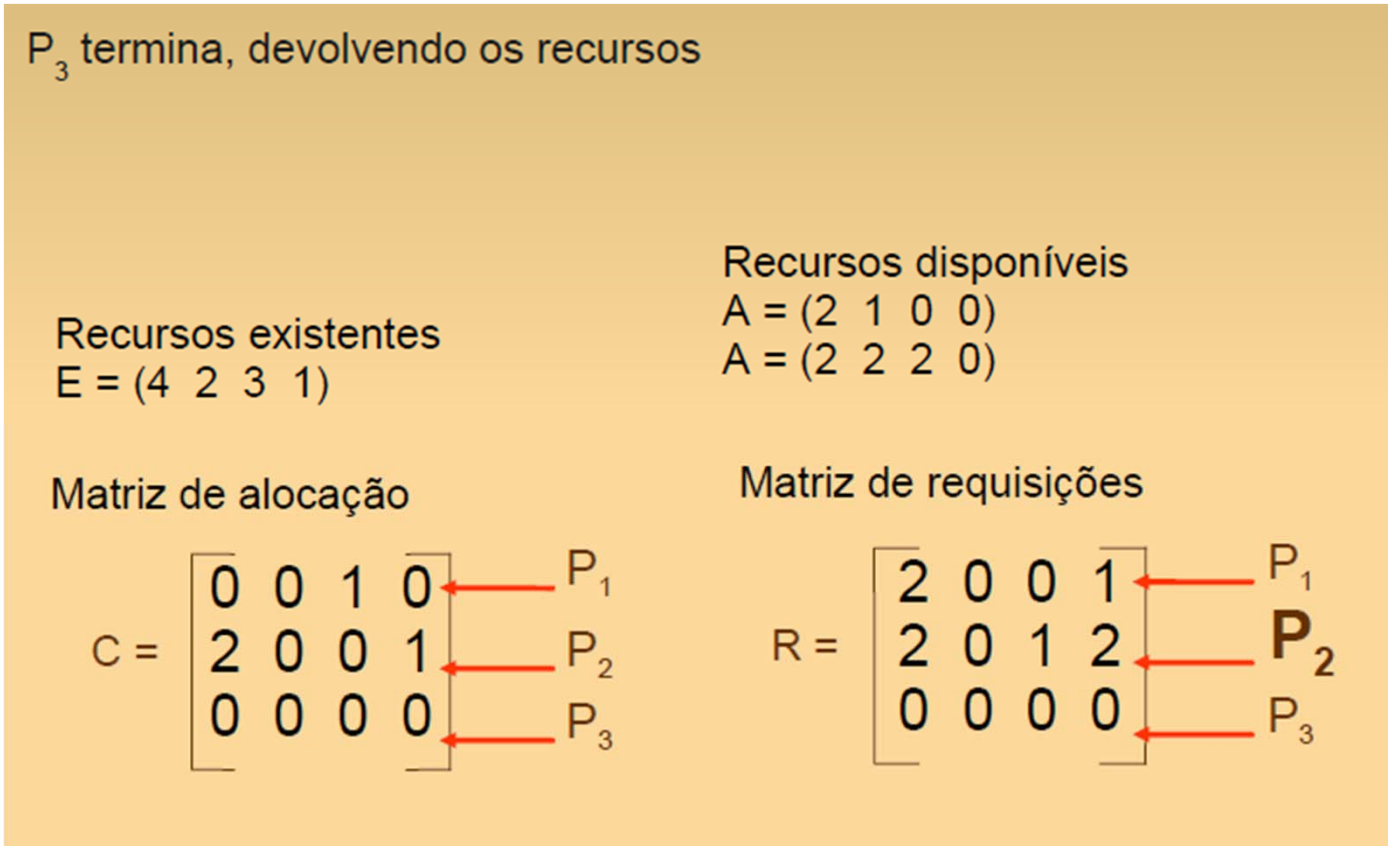
$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 2 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 2 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow \mathbf{P_2} \\ \leftarrow P_3 \end{matrix}$$

Estruturas de dados necessárias ao algoritmo de detecção de deadlock (2º Caso)

Detecção com um Recurso de Cada Tipo (3)



Estruturas de dados necessárias ao algoritmo de detecção de deadlock (2º Caso)

Detecção com um Recurso de Cada Tipo (3)

Recursos existentes
E = (4 2 3 1)

Recursos disponíveis
A = (2 1 0 0)
A = (2 2 2 0)

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow P_2 \\ \leftarrow P_3 \end{matrix}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ \mathbf{2} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow P_1 \\ \leftarrow \mathbf{P_2} \\ \leftarrow P_3 \end{matrix}$$

Nessa situação, nenhum processo pode ser atendido!
DEADLOCK

Deadlock ocorrerá se não pudermos mais distribuir recursos disponíveis aos processos requisitantes (em R)

Estruturas de dados necessárias ao algoritmo de detecção de deadlock (2º Caso)

Recuperação de Deadlock (1)

- Recuperação através de preempção
  - retirar um recurso de algum outro processo
  - depende da natureza do recurso
- Recuperação através de reversão de estado
  - verifica um processo periodicamente
  - usa este estado salvo
  - reinicia o processo se este é encontrado em estado de deadlock

Recuperação de Deadlock (2)

- Recuperação através da eliminação de processos
  - forma mais grosseira mas também mais simples de quebrar um deadlock
  - elimina um dos processos no ciclo de deadlock
  - os outros processos conseguem seus recursos
  - escolhe processo que pode ser reexecutado desde seu início

Evitando Deadlocks

- Evitar dinamicamente o problema:
  - Alocação individual de recursos são normalmente feitos à medida que o processo necessita;
  - Escalonamento cuidadoso → alto custo;
    - Conhecimento prévio dos recursos que serão utilizados;
  - Algoritmos:
    - Banqueiro para um único tipo de recurso;
    - Banqueiro para vários tipos de recursos;
  - Usam a noção de Estados Seguros e Inseguros;

Evitando Deadlocks

- Evitar dinamicamente o problema:
  - Estados seguros: não provocam deadlocks e há uma maneira de atender a todas as requisições pendentes finalizando normalmente todos os processos;
    - A partir de um estado seguro, existe a garantia de que os processos terminarão;
  - Estados inseguros: podem provocar deadlocks, mas não necessariamente provocam;
    - A partir de um estado inseguro, não é possível garantir que os processos terminarão corretamente;

Estados Seguros e Inseguros (1)

	Possui máx.	
A	3	9
B	2	4
C	2	7

Disponível: 3
(a)

	Possui máx.	
A	3	9
B	4	4
C	2	7

Disponível: 1
(b)

	Possui máx.	
A	3	9
B	0	–
C	2	7

Disponível: 5
(c)

	Possui máx.	
A	3	9
B	0	–
C	7	7

Disponível: 0
(d)

	Possui máx.	
A	3	9
B	0	–
C	0	–

Disponível: 7
(e)

Demonstração de que o estado em (a) é seguro

Estados Seguros e Inseguros (2)

	Possui máx.	
A	3	9
B	2	4
C	2	7

Disponível: 3
(a)

	Possui máx.	
A	4	9
B	2	4
C	2	7

Disponível: 2
(b)

	Possui máx.	
A	4	9
B	4	4
C	2	7

Disponível: 0
(c)

	Possui máx.	
A	4	9
B	–	–
C	2	7

Disponível: 4
(d)

Demonstração de que o estado em (b) é inseguro

O Algoritmo do Banqueiro para um Único Recurso

	Possui	máx.
A	0	6
B	0	5
C	0	4
D	0	7

Disponível: 10

(a)

	Possui	máx.
A	1	6
B	1	5
C	2	4
D	4	7

Disponível: 2

(b)

	Possui	máx.
A	1	6
B	2	5
C	2	4
D	4	7

Disponível: 1

(c)

Três estados de alocação de recursos

a) seguro
b) seguro
c) inseguro

O Algoritmo do Banqueiro para Múltiplos Recursos

Processo	Unidades de fita	Plotters	Scanners	Unidades de CD-ROM
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

Recursos alocados

Processo	Unidades de fita	Plotters	Scanners	Unidades de CD-ROM
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

Recursos ainda necessários

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

Exemplo do algoritmo do banqueiro com múltiplos recursos

Prevenção de Deadlock

Atacando a Condição de Exclusão Mútua

- Alguns dispositivos (como uma impressora) podem fazer uso de *spool*
  - o daemon de impressão é o único que usa o recurso impressora
  - desta forma deadlock envolvendo a impressora é eliminado
- Nem todos os dispositivos podem fazer uso de *spool*
- Princípio:
  - evitar alocar um recurso quando ele não for absolutamente necessário necessário
  - tentar assegurar que o menor número possível de processos possa de fato requisitar o recurso

Prevenção de Deadlock

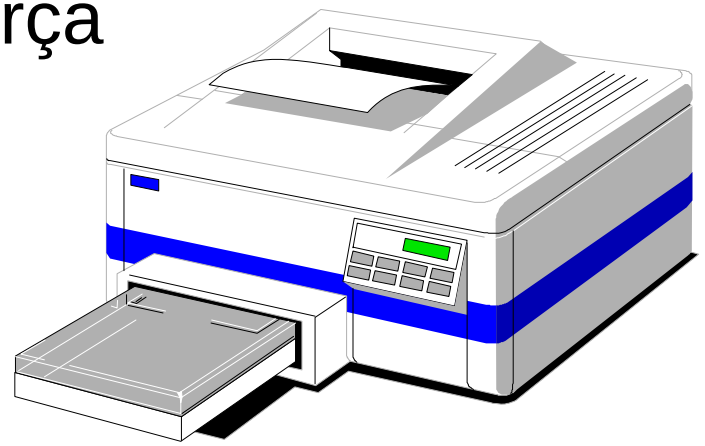
Atacando a Condição de Posse e Espera

- Exigir que todos os processos requisitem os recursos antes de iniciarem
  - um processo nunca tem que esperar por aquilo que precisa
- Problemas
  - podem não saber quantos e quais recursos vão precisar no início da execução
  - e também retêm recursos que outros processos poderiam estar usando
- Variação:
  - processo deve desistir de todos os recursos
  - para então requisitar todos os que são imediatamente necessários

Prevenção de Deadlock

Atacando a Condição de Não Preempção

- Esta é uma opção inviável
- Considere um processo de posse de uma impressora
  - no meio da impressão
  - retoma a impressora a força
  - !!??

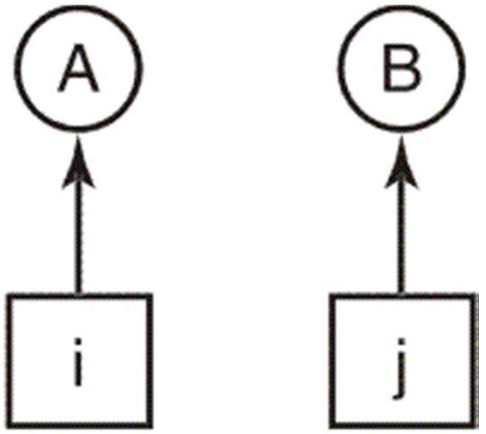


Prevenção de Deadlock

Atacando a Condição de Espera Circular (1)

1. Imagesetter
2. Scanner
3. Plotter
4. Unidade de fita
5. Unidade de CD-ROM

(a)



(b)

a) Recursos ordenados numericamente

b) Um grafo de recursos

Prevenção de Deadlock

Atacando a Condição de Espera Circular (2)

Condição	Abordagem contra deadlocks
Exclusão mútua	Usar spool em tudo
Posse-e-espera	Requisitar inicialmente todos os recursos necessários
Não preempção	Retomar os recursos alocados
Espera circular	Ordenar numericamente os recursos

Resumo das abordagens para prevenir deadlock

Condição de Inanição - *Starvation*

- Algoritmo para alocar um recurso
  - pode ser ceder para o *job mais curto primeiro*
- Funciona bem para múltiplos jobs curtos em um sistema
- Jobs longos podem ser preteridos indefinidamente
  - mesmo não estando bloqueados
- solução:
  - política do *primeiro a chegar, primeiro a ser servido*

Exercício

Verificar se o estado é seguro ou inseguro:

- Caso seguro: Propor a possível resolução
- Caso Inseguro: Apresentar onde se daria o deadlock

i)Sistema A tem 12 dispositivos; apenas 1 está disponível.

Número do programa	Dispositivos alocados	Máximo de requisições	Requisições restantes
1	5	6	1
2	4	7	3
3	2	6	2
4	0	2	2

Exercício

Verificar se o estado é seguro ou inseguro:

- Caso seguro: Propor a possível resolução
- Caso Inseguro: Apresentar onde se daria o deadlock

ii)Sistema B tem 14 dispositivos; apenas 2 está disponível.

Número do programa	Dispositivos alocados	Máximo de requisições	Requisições restantes
1	5	8	**2**
2	3	9	**6**
3	4	8	**3**

Exercício

Verificar se o estado é seguro ou inseguro:

- Caso seguro: Propor a possível resolução
- Caso Inseguro: Apresentar onde se daria o deadlock

iii) Sistema C tem 10 dispositivos; apenas 2 está disponível.

Número do programa	Dispositivos alocados	Máximo de requisições	Requisições restantes
1	3	8	2
2	3	9	6
3	2	8	6