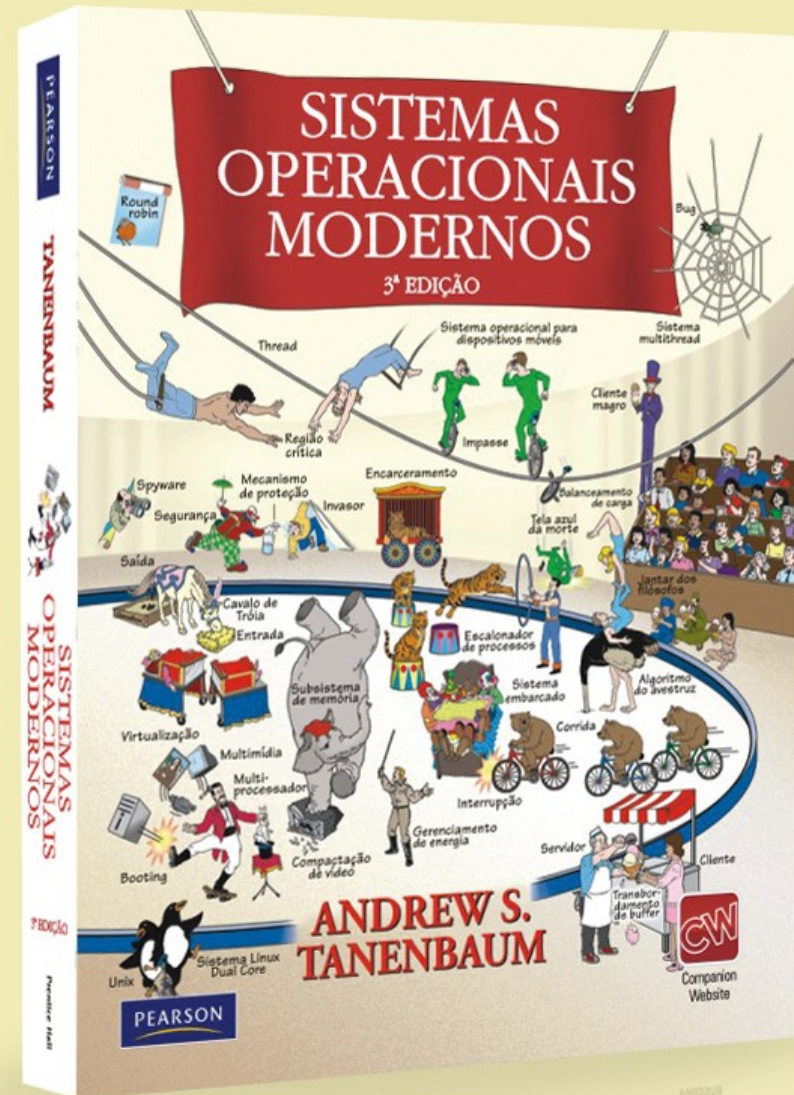


Sistemas operacionais
modernos
Terceira edição
ANDREW S. TANENBAUM

Capítulo 6
Impasses





Recursos preemptíveis e não preemptíveis

A sequência de eventos necessária ao uso de um determinado recurso é dada abaixo de maneira abstrata:

1. Requisitar o recurso.
2. Usar o recurso.
3. Liberar o recurso.

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

Aquisição de recursos

```
typedef int semaphore;  
semaphore resource_1;
```

```
void process_A(void) {  
    down(&resource_1);  
    use_resource_1( );  
    up(&resource_1);  
}
```

(a)

```
typedef int semaphore;  
semaphore resource_1;  
semaphore resource_2;
```

```
void process_A(void) {  
    down(&resource_1);  
    down(&resource_2);  
    use_both_resources( );  
    up(&resource_2);  
    up(&resource_1);  
}
```

(b)

Figura 6.1 O uso de um semáforo para proteger recursos.

(a) Um recurso. (b) Dois recursos.

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

```
typedef int semaphore;
```

```
semaphore resource_1;  
semaphore resource_2;
```

```
void process_A(void) {  
    down(&resource_1);  
    down(&resource_2);  
    use_both_resources( );  
    up(&resource_2);  
    up(&resource_1);  
}
```

```
void process_B(void) {  
    down(&resource_1);  
    down(&resource_2);  
    use_both_resources( );  
    up(&resource_2);  
    up(&resource_1);  
}
```

(a)

```
semaphore resource_1;  
semaphore resource_2;
```

```
void process_A(void) {  
    down(&resource_1);  
    down(&resource_2);  
    use_both_resources( );  
    up(&resource_2);  
    up(&resource_1);  
}
```

```
void process_B(void) {  
    down(&resource_2);  
    down(&resource_1);  
    use_both_resources( );  
    up(&resource_1);  
    up(&resource_2);  
}
```

(b)

Figura 6.2 (a) Código sem impasse. (b) Código com possibilidade de impasse.



Introdução aos impasses

Um impasse pode ser definido formalmente como:

Um conjunto de processos estará em situação de impasse se todo processo pertencente ao conjunto estiver esperando por um evento que somente outro processo desse mesmo conjunto poderá fazer acontecer.

Condições para impasse de recursos

1. Condição de exclusão mútua.
2. Condição de posse e espera.
3. Condição de não preempção.
4. Condição de espera circular.

Modelagem de impasses

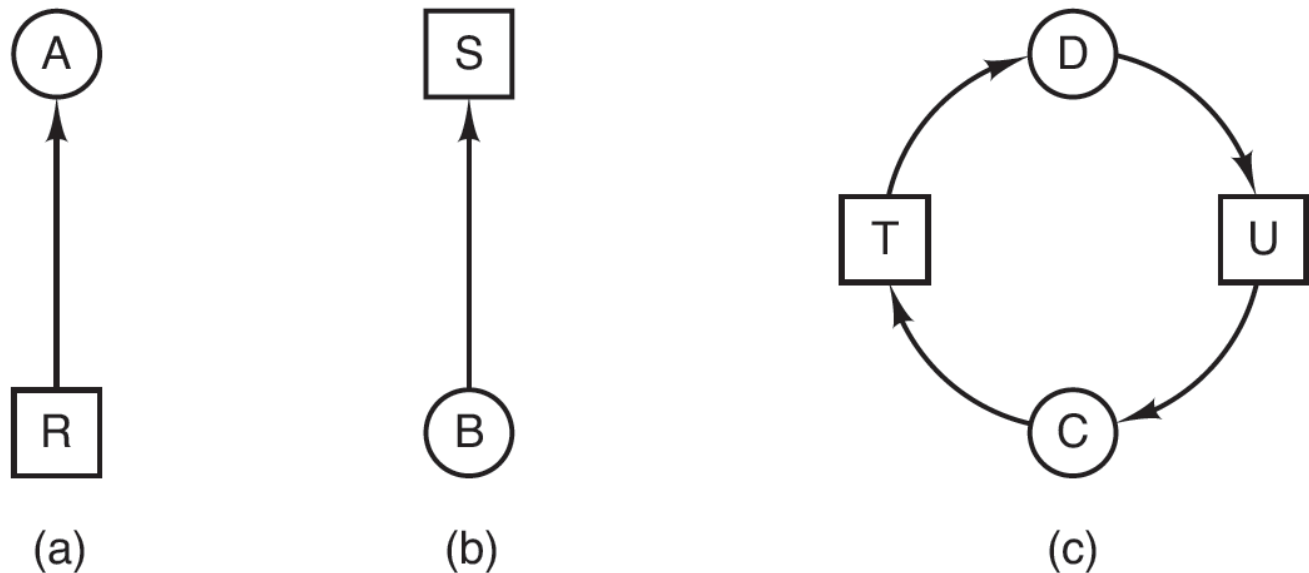


Figura 6.3 Grafos de alocação de recursos. (a) Processo de posse de um recurso. (b) Processo requisitando um recurso. (c) Impasse.

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

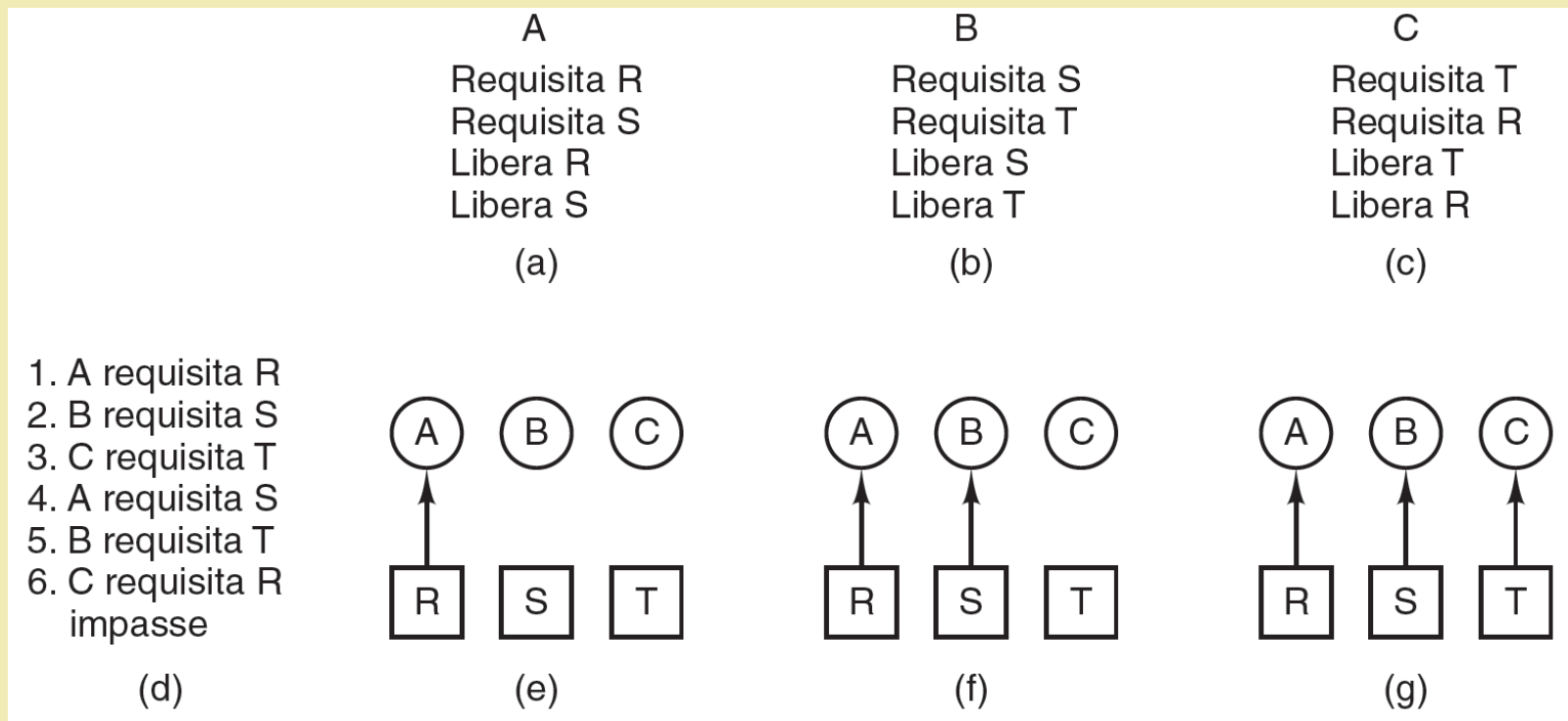


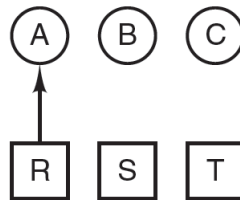
Figura 6.4 Exemplo de como um impasse ocorre e como pode ser evitado.

SISTEMAS OPERACIONAIS MODERNOS

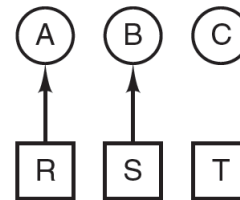
3ª EDIÇÃO

1. A requisita R
2. B requisita S
3. C requisita T
4. A requisita S
5. B requisita T
6. C requisita R
impasse

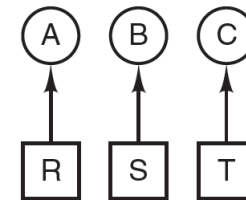
(d)



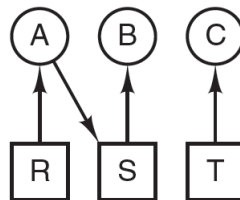
(e)



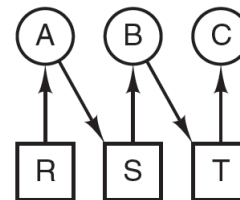
(f)



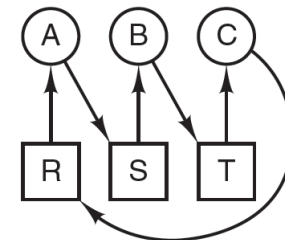
(g)



(h)



(i)



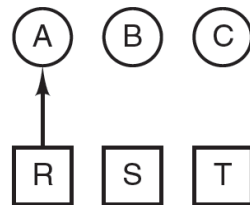
(j)

Figura 6.4 Exemplo de como um impasse ocorre e como pode ser evitado.

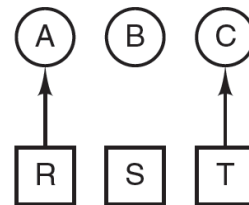
SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

1. A requisita R
 2. C requisita T
 3. A requisita S
 4. C requisita R
 5. A libera R
 6. A libera S
- nenhum impasse

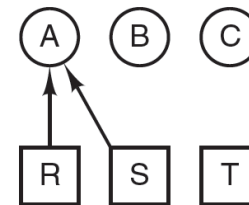


(k)

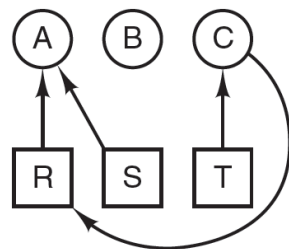


(l)

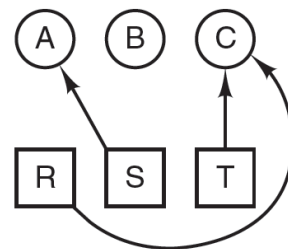
(m)



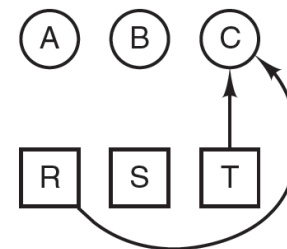
(n)



(o)



(p)



(q)

Figura 6.4 Exemplo de como um impasse ocorre e como pode ser evitado.

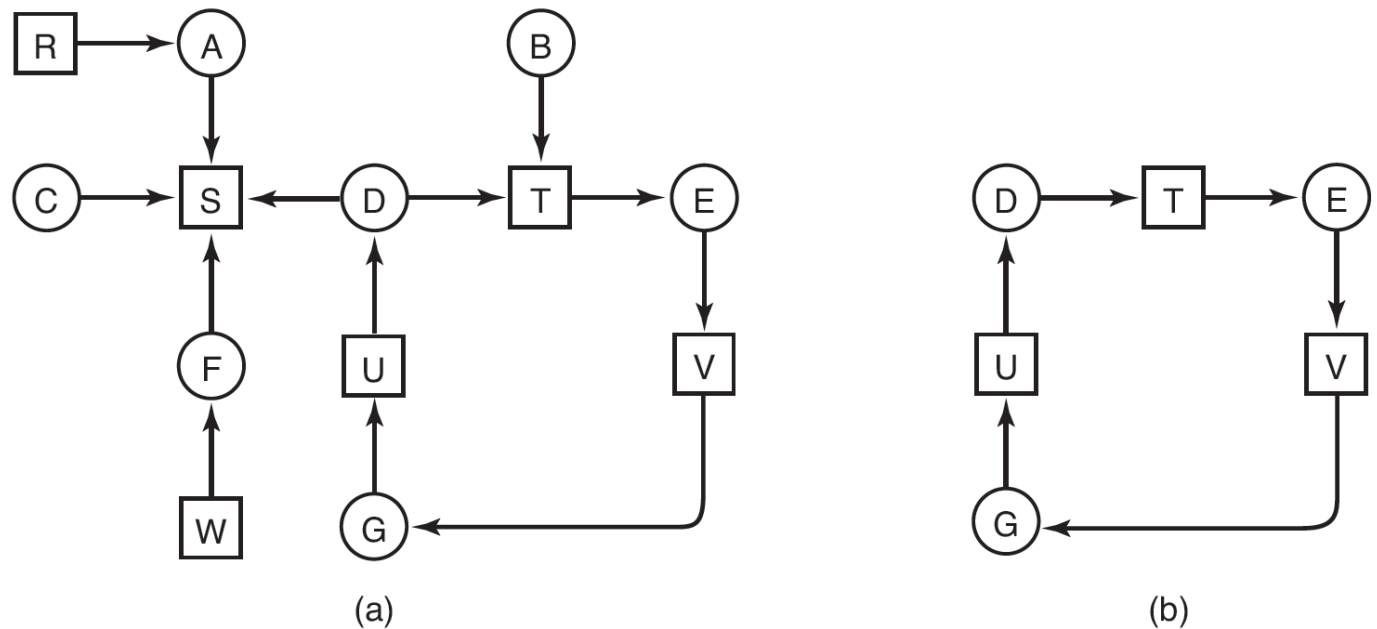
SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

Estratégias para lidar com impasses:

1. Ignorar o problema.
2. Detecção e recuperação. Deixe os impasses ocorrerem, detecte e recupere.
3. Anulação dinâmica por meio de alocação cuidadosa de recursos.
4. Prevenção, negando estruturalmente uma das quatro condições necessárias.

Detecção de impasse com um recurso de cada tipo



■ **Figura 6.5** (a) Um gráfico de recursos. (b) Um ciclo extraído de (a).



Algoritmo para detecção de impasse:

1. Para cada nó, N no gráfico, realize os cinco passos seguintes , usando N como nó inicial.
2. Inicialize L para uma lista vazia e assinale todos os arcos como demarcados.
3. Insira o nó atual no final de L, verifique se o nó aparece em L duas vezes. Se sim, o gráfico contém um ciclo (listado em L) e o algoritmo termina.

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

4. A partir do nó dado, verifique se existe algum arco de saída desmarcado. Em caso afirmativo, vá para o passo 5; do contrário, vá para o passo 6.
5. Escolha aleatoriamente um arco de saída desmarcado e marque-o. Então, siga esse arco para obter o novo nó atual e vá para o passo 3.
6. Se esse nó for o inicial, o gráfico não conterá ciclo algum e o algoritmo terminará. Senão, o final foi alcançado. Remova-o e volte para o nó anterior — isto é, aquele que era atual antes desse —, marque-o como atual e vá para o passo 3.


SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

Detecção de impasse com múltiplos recursos de cada tipo

Recursos existentes
($E_1, E_2, E_3, \dots, E_m$)

Matriz de alocação atual




C_{11}	C_{12}	C_{13}	\dots	C_{1m}
C_{21}	C_{22}	C_{23}	\dots	C_{2m}
\vdots	\vdots	\vdots		\vdots
C_{n1}	C_{n2}	C_{n3}	\dots	C_{nm}

Linha n é a alocação atual para o processo n

Recursos disponíveis
($A_1, A_2, A_3, \dots, A_m$)

Matriz de requisições



R_{11}	R_{12}	R_{13}	\dots	R_{1m}
R_{21}	R_{22}	R_{23}	\dots	R_{2m}
\vdots	\vdots	\vdots		\vdots
R_{n1}	R_{n2}	R_{n3}	\dots	R_{nm}

Linha 2 informa qual é a necessidade do processo 2

Figura 6.6 As quatro estruturas de dados necessárias ao algoritmo de detecção de impasses.

SISTEMAS OPERACIONAIS MODERNOS

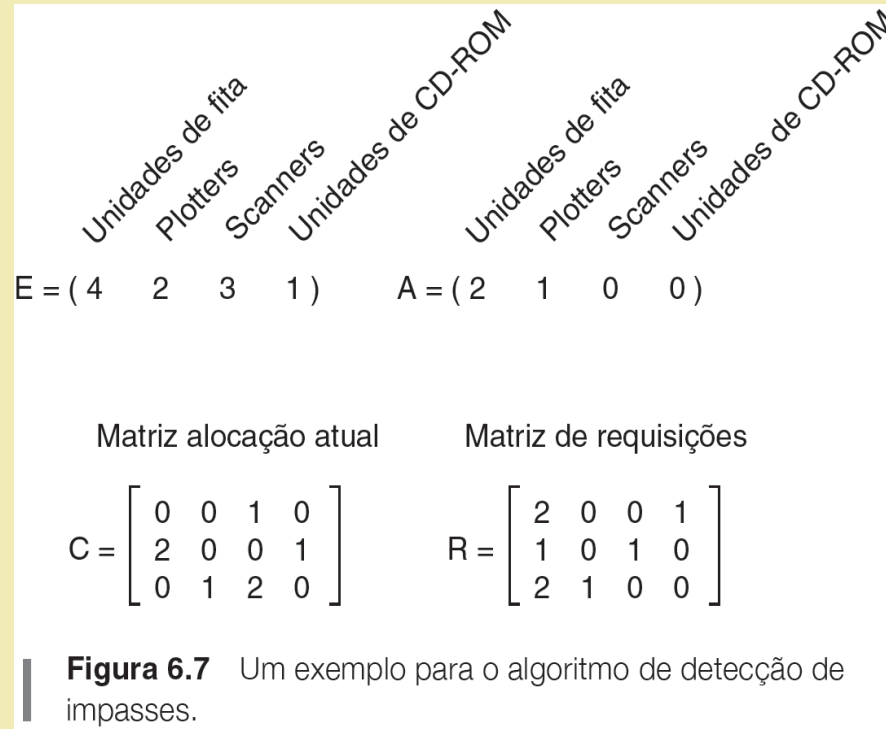
3ª EDIÇÃO

Algoritmo de detecção de impasse:

1. Procure um processo desmarcado, P_i , para o qual a i -ésima linha de R seja menor ou igual à correspondente de A .
2. Se esse processo for encontrado, adicione a i -ésima linha de C à correspondente de A , marque o processo e volte para o passo 1.
3. Se não existir esse processo, o algoritmo terminará.

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO





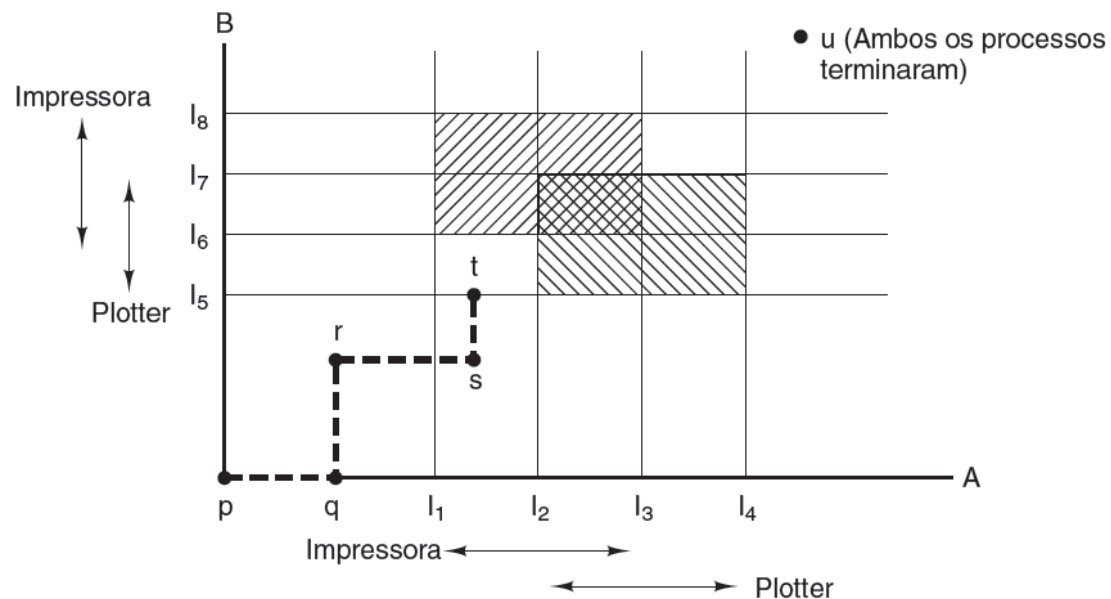
Recuperação de impasses

- Recuperação por preempção.
- Recuperação por retrocesso.
- Recuperação por eliminação de processos.

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

Evitando impasses



■ **Figura 6.8** A trajetória de recursos de dois processos.

Estados seguros e inseguros

Possui máx.	Possui máx.	Possui máx.	Possui máx.	Possui máx.																																													
<table><tr><td>A</td><td>3</td><td>9</td></tr><tr><td>B</td><td>2</td><td>4</td></tr><tr><td>C</td><td>2</td><td>7</td></tr></table>	A	3	9	B	2	4	C	2	7	<table><tr><td>A</td><td>3</td><td>9</td></tr><tr><td>B</td><td>4</td><td>4</td></tr><tr><td>C</td><td>2</td><td>7</td></tr></table>	A	3	9	B	4	4	C	2	7	<table><tr><td>A</td><td>3</td><td>9</td></tr><tr><td>B</td><td>0</td><td>–</td></tr><tr><td>C</td><td>2</td><td>7</td></tr></table>	A	3	9	B	0	–	C	2	7	<table><tr><td>A</td><td>3</td><td>9</td></tr><tr><td>B</td><td>0</td><td>–</td></tr><tr><td>C</td><td>7</td><td>7</td></tr></table>	A	3	9	B	0	–	C	7	7	<table><tr><td>A</td><td>3</td><td>9</td></tr><tr><td>B</td><td>0</td><td>–</td></tr><tr><td>C</td><td>0</td><td>–</td></tr></table>	A	3	9	B	0	–	C	0	–
A	3	9																																															
B	2	4																																															
C	2	7																																															
A	3	9																																															
B	4	4																																															
C	2	7																																															
A	3	9																																															
B	0	–																																															
C	2	7																																															
A	3	9																																															
B	0	–																																															
C	7	7																																															
A	3	9																																															
B	0	–																																															
C	0	–																																															
Disponível: 3	Disponível: 1	Disponível: 5	Disponível: 0	Disponível: 7																																													
(a)	(b)	(c)	(d)	(e)																																													

■ **Figura 6.9** Demonstração de que o estado em (a) é seguro.

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

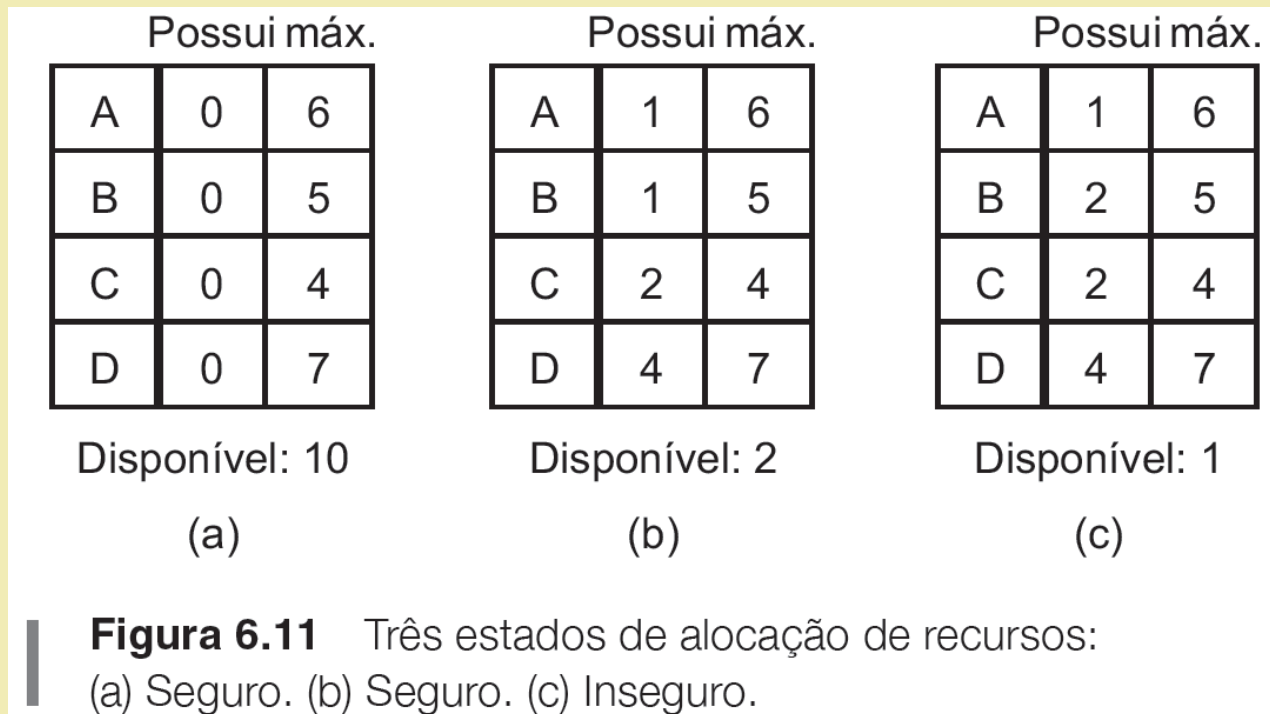
Possui máx.			Possui máx.			Possui máx.			Possui máx.		
A	3	9	A	4	9	A	4	9	A	4	9
B	2	4	B	2	4	B	4	4	B	–	–
C	2	7	C	2	7	C	2	7	C	2	7
Disponível: 3			Disponível: 2			Disponível: 0			Disponível: 4		
(a)			(b)			(c)			(d)		

■ **Figura 6.10** Demonstração de que o estado em (b) é inseguro.

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

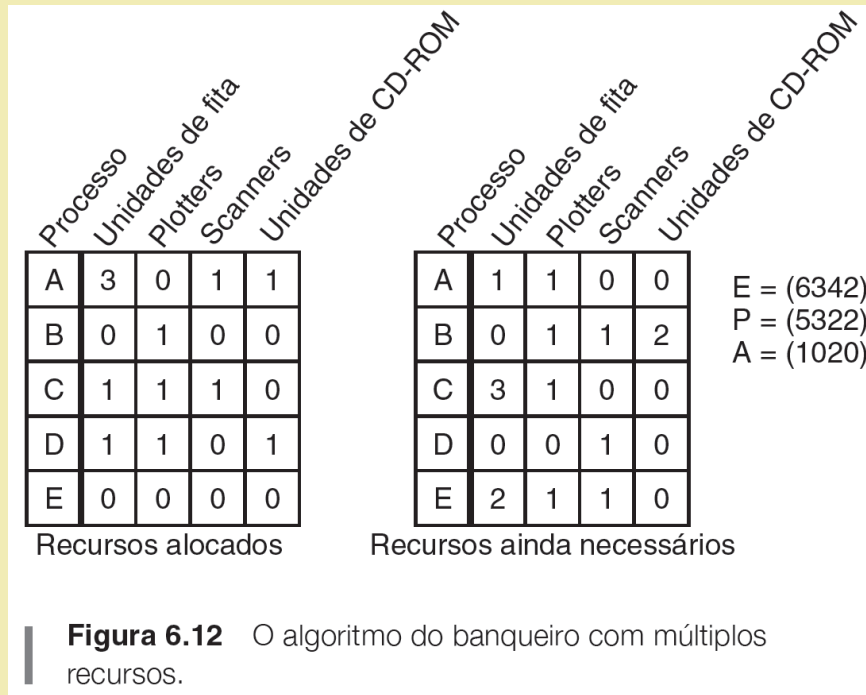
Algoritmo do banqueiro para um único recurso



SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

Algoritmo do banqueiro para múltiplos recursos



SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

O algoritmo que verifica se um estado é seguro é descrito a seguir:

1. Procure uma linha R, cujas necessidades de recursos sejam $\leq A$. Se não existir tal linha o sistema acabará por entrar em impasse, já que nenhum processo poderá ser executado por completo.
2. Considere que o processo da linha escolhida requer todos os recursos que necessita e termina. Marque o processo como terminado, acrescente todos os seus recursos ao vetor A.
3. Repita os passos 1 e 2 até que todos os processos estejam marcados como terminados (estado seguro) ou até que não haja processo cujo recurso precise ser encontrado (há um impasse).

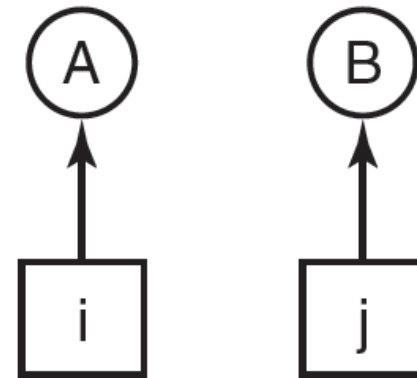
Prevenção de impasses

- Atacando a condição de exclusão mútua.
- Atacando a condição de posse e espera.
- Atacando a condição de preempção.
- Atacando a condição de espera circular.

Atacando a condição de espera circular

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

(a)



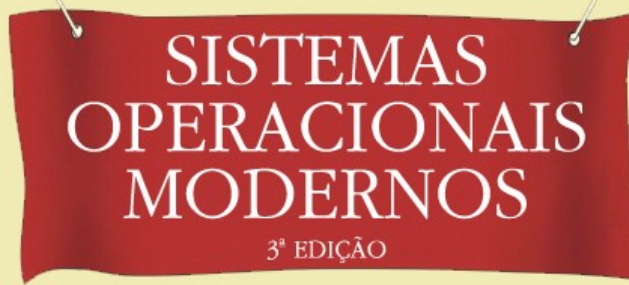
(b)

Figura 6.13 (a) Recursos ordenados numericamente. (b) Um gráfico de recursos.

Abordagens contra impasses

Condição	Abordagem contra impasses
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

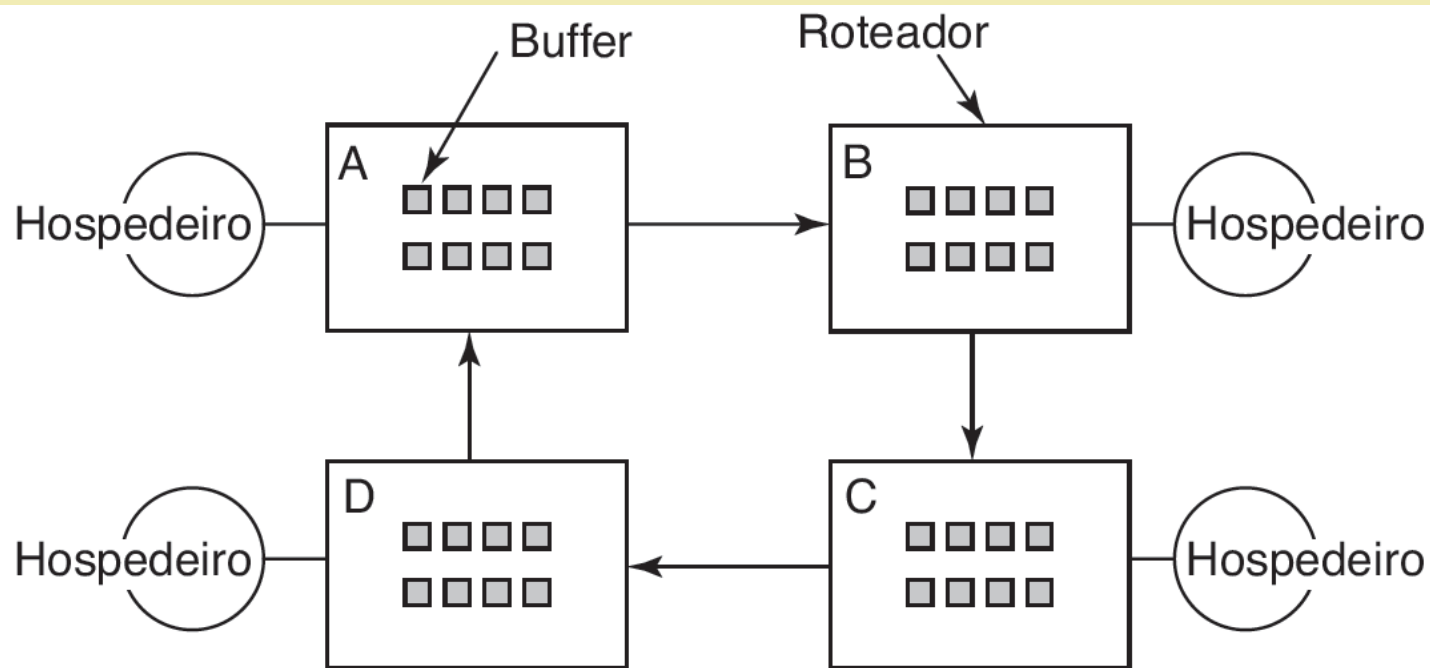
■ **Tabela 6.1** Resumo das abordagens para prevenir impasses.



Outras questões

- Bloqueio em duas fases.
- Impasses de comunicação.
- Livelock.
- Condição de inanição.

Impasses de comunicação



■ **Figura 6.14** Um impasse de recurso em uma rede.

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

Livelock

```
void process_A(void) {  
    enter_region(&resource_1);  
    enter_region(&resource_2);  
    use_both_resources( );  
    leave_region(&resource_2);  
    leave_region(&resource_1);  
}  
  
void process_B(void) {  
    enter_region(&resource_2);  
    enter_region(&resource_1);  
    use_both_resources( );  
    leave_region(&resource_1);  
    leave_region(&resource_2);  
}
```

■ **Figura 6.15** A espera ocupada que pode acarretar um livelock.