

Aula 11

Sistemas Operacionais I

Deadlocks

Prof. Julio Cezar Estrella
jcezar@icmc.usp.br

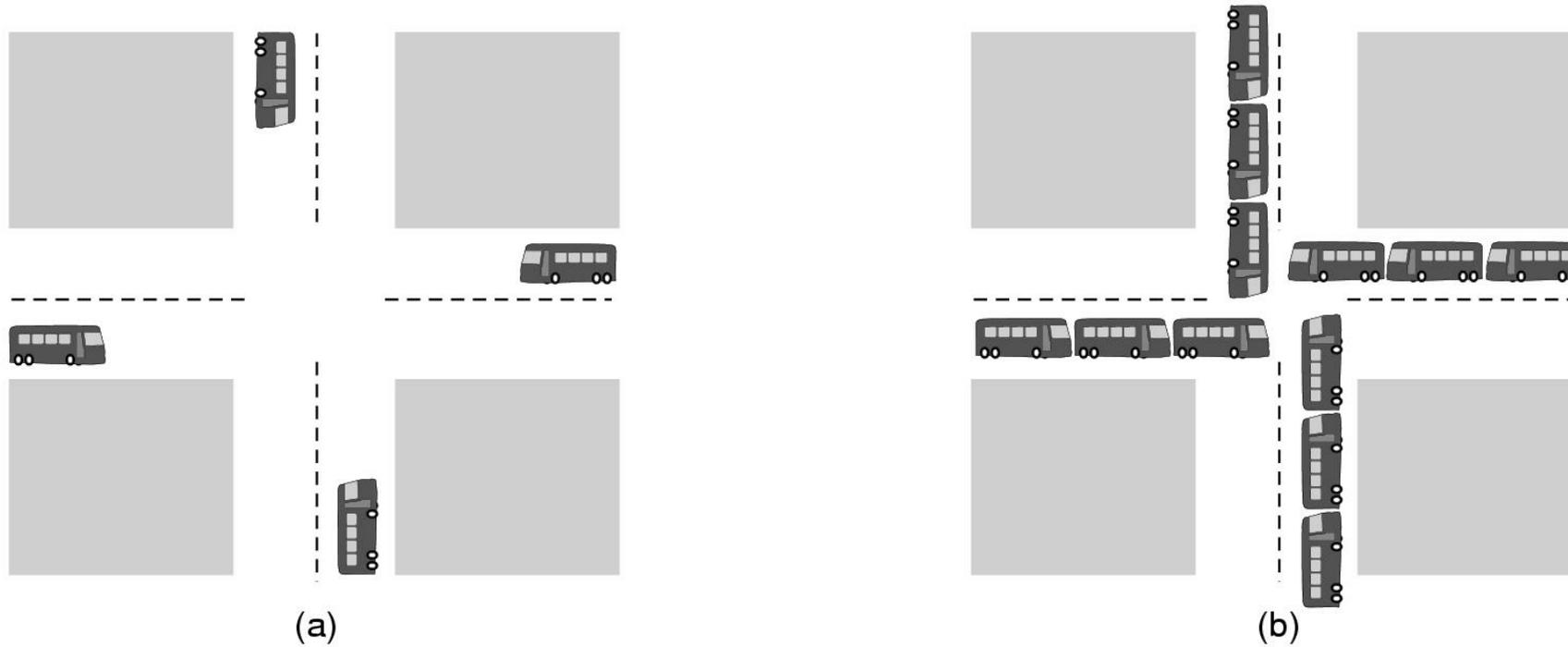
*Material adaptado de
Sarita Mazzini Bruschi*

baseados no livro Sistemas Operacionais Modernos de A. Tanenbaum

Deadlocks

- Dispositivos e recursos são compartilhados a todo momento: impressora, disco, arquivos, etc...;
- *Deadlock*: processos ficam parados sem possibilidade de poderem continuar seu processamento;

Deadlocks



Uma situação de *deadlock*

Deadlocks



Deadlocks



Recursos

- Recursos: objetos acessados, os quais podem ser tanto hardware quanto uma informação
 - Preemptivos: podem ser retirados do processo sem prejuízos;
 - Memória;
 - CPU;
 - Não-preemptivos: não podem ser retirados do processo, pois causam prejuízos;
 - CD-ROM;
 - Unidades de fita;
 - *Deadlocks* ocorrem com recursos não-preemptivos;

Recursos

- Operações sobre recursos/dispositivos:
 - Requisição do recurso;
 - Utilização do recurso;
 - Liberação do recurso;
- Se o recurso requerido não está disponível, duas situações podem ocorrer:
 - Processo que requisitou o recurso fica bloqueado até que o recurso seja liberado, ou;
 - Processo que requisitou o recurso falha, e depois de um certo tempo tenta novamente requisitar o recurso;

Recursos

- Aquisição do recurso
 - Para alguns tipos de recursos, os processos dos usuários gerenciam o uso dos recursos, através, por exemplo, de semáforos
 - Exemplo: acesso a registros em um sistema de banco de dados
- Se vários processos tentam acessar os mesmos recursos, podem ocorrer situações onde a ordem de solicitação dos recursos pode conduzir a um deadlock ou não

Deadlocks

Potencial *deadlock*

```
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);}
```

Livre de *deadlock*

```
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_1);
    up(&resource_2);}
```

Deadlocks

- Definição formal:
 - “Um conjunto de processos estará 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.”

Deadlocks

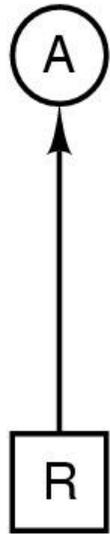
- Quatro condições para que ocorra um *deadlock*:
 - Exclusão mútua: cada recurso pode estar somente em uma de duas situações: ou associado a um único processo ou disponível;
 - Posse e espera (*hold and wait*): processos que já possuem algum recurso podem requer outros recursos;
 - Não-preempção: recursos já alocados não podem ser retirados do processo que os alocou; somente o processo que alocou os recursos pode liberá-los;
 - Espera Circular: um processo pode esperar por recursos alocados a outro processo;
- Todas as condições devem ocorrer para que ocorra um *deadlock*

Deadlocks

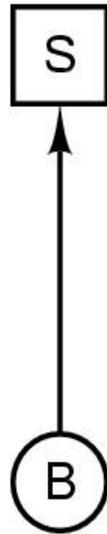
- Geralmente, *deadlocks* são representados por grafos a fim de facilitar sua detecção, prevenção e recuperação
 - Ocorrência de ciclos pode levar a um *deadlock*;

Deadlocks

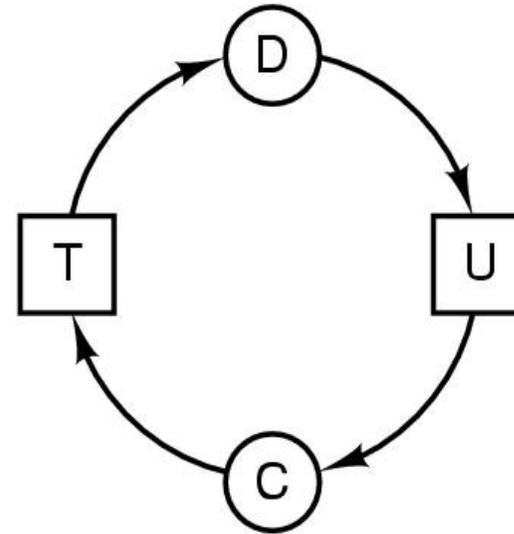
Grafos de alocação de recursos



(a)



(b)



(c)

a) Recurso R alocado ao **Processo A**

b) Processo B requisita **Recurso S**

c) *Deadlock*

Deadlocks

- Quatro estratégias para tratar *deadlocks*:
 1. Ignorar o problema;
 2. Detectar e recuperar o problema;
 3. Evitar dinamicamente o problema – alocação cuidadosa de recursos;
 4. Prevenir o problema por meio da não satisfação de uma das quatro condições citadas anteriormente;

Deadlocks

- **(1) Ignorar o problema:**
 - Frequência do problema;
 - Alto custo – estabelecimento de condições para o uso de recursos;
 - UNIX e WINDOWS;
 - Algoritmo do AVESTRUZ;



Deadlocks

- **(2) Detectar e Recuperar o problema:**

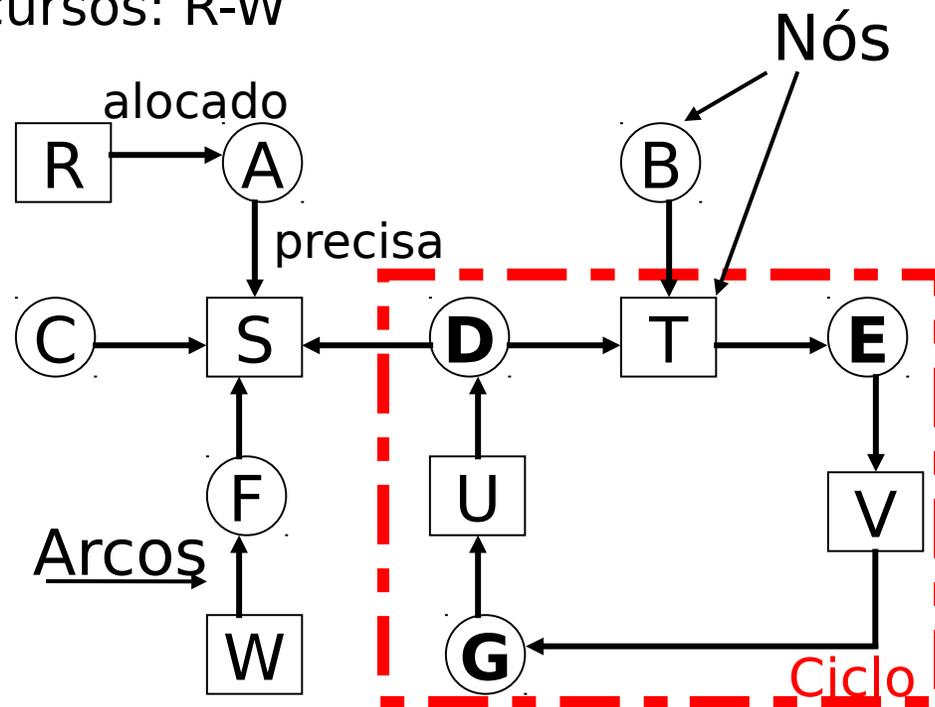
- Processos estão com todos os recursos alocados;
- Procedimento: Permite que os *deadlocks* ocorram, tenta detectar as causas e solucionar a situação;
- Algoritmos:
 - Detecção com um recurso de cada tipo;
 - Detecção com vários recursos de cada tipo;
 - Recuperação por meio de preempção;
 - Recuperação por meio de *rollback* (volta ao passado);
 - Recuperação por meio de eliminação de processos;

Deadlocks

- Detecção com um recurso de cada tipo:
 - Construção de um grafo;
 - Se houver ciclos, existem potenciais *deadlocks*;

Processos: A-G

Recursos: R-W



Situação:

PA usa R e precisa de S;
PB precisa de T;
PC precisa de S;
PD usa U e precisa de S e T;
PE usa T e precisa de V;
PF usa W e precisa de S;
PG usa V e precisa de U;

Pergunta:

Há possibilidade de *deadlock*?

Deadlocks

- Detecção com vários recursos de cada tipo:
 - Classes diferentes de recursos – vetor de recursos existentes (E):
 - Se classe1=unidade de fita e $E_1=2$, então existem duas unidades de fita;
 - Vetor de recursos disponíveis (A):
 - Se ambas as unidades de fita estiverem alocadas, $A_1=0$;
 - Duas matrizes:
 - C: matriz de alocação corrente;
 - C_{ij} : número de instâncias do recurso j entregues ao processo i;
 - R: matriz de requisições;
 - R_{ij} : número de instâncias do recurso j que o processo i precisa;

Deadlocks

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

Três processos já usam:

P_1 usa um *scanner*;

P_2 usa duas unidades de fita e uma de CD-ROM;

P_3 usa um *plotter* e dois *scanners*;

Requisições:

P_1 requisita duas unidades de fita e um CD-ROM;

P_2 requisita uma unidade de fita e um *scanner*;

P_3 requisita duas unidades de fita e um *plotter*;

Recursos existentes

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix}$$

UF P S UCD

Recursos disponíveis

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

UF P S UCD

Matriz de alocação

$$C = \begin{array}{c} \begin{array}{cccc} \text{UF} & \text{P} & \text{S} & \text{UCD} \\ \hline 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{array} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array} \end{array}$$

Recursos

Matriz de requisições

$$R = \begin{array}{c} \begin{array}{cccc} \text{UF} & \text{P} & \text{S} & \text{UCD} \\ \hline 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{array} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array} \end{array}$$

Deadlocks

4 unidades de fita;
 2 *plotters*;
 3 *scanners*;
 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 um *scanner*;
 P_3 requisita duas unidades de fita e um *plotter*;

Recursos existentes Recursos disponíveis

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

$$A = (2 \ 1 \ 0 \ 0) \quad \mathbf{P_3 \ pode \ executar}$$

$$A = (\mathbf{0 \ 0 \ 0 \ 0}) \quad \mathbf{P_3 \ alocou \ recursos}$$

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ \mathbf{2} & \mathbf{2} & \mathbf{2} & \mathbf{0} \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow \mathbf{P_3} \end{array}$$

Deadlocks

4 unidades de fita;
 2 *plotter*;
 3 *scanners*;
 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 um *scanner*;
 P_3 requisita duas unidades de fita e um *plotter*;

Recursos existentes Recursos disponíveis

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

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

$$A = (2 \ 2 \ 2 \ 0) \quad \mathbf{P_3 \ liberou \ recursos}$$

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow \mathbf{P_3} \end{array}$$

Deadlocks

4 unidades de fita;
 2 *plotter*;
 3 *scanners*;
 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 um *scanner*;
 P_3 requisita duas unidades de fita e um *plotter*;

Recursos existentes Recursos disponíveis

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

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

$$A = (2 \ 2 \ 2 \ 0) \mathbf{P}_2 \text{ pode executar}$$

$$A = (\mathbf{1} \ \mathbf{2} \ \mathbf{1} \ \mathbf{0}) \mathbf{P}_2 \text{ alocou recursos}$$

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ \mathbf{3} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow \mathbf{P}_2 \\ \longleftarrow P_3 \end{array}$$

Deadlocks

4 unidades de fita;
 2 *plotter*;
 3 *scanners*;
 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 um *scanner*;

P_3 requisita duas unidades de fita e um *plotter*;

Recursos existentes Recursos disponíveis

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

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

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

$$A = (\mathbf{4} \ \mathbf{2} \ \mathbf{2} \ \mathbf{1}) \ \mathbf{P_2 \ liberou \ recursos}$$

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow \mathbf{P_2} \\ \longleftarrow P_3 \end{array}$$

Deadlocks

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

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

Requisições:

P_1 requisita duas unidades de fita e um CD-ROM;

P_2 requisita uma unidade de fita e um *scanner*;

P_3 requisita duas unidades de fita e um *plotter*;

Recursos disponíveis

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

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

$$A = (4 \ 2 \ 2 \ 1) \quad \mathbf{P_1 \ pode \ executar}$$

$$A = (2 \ 2 \ 2 \ 0) \quad \mathbf{P_1 \ alocou \ recursos}$$

Matriz de alocação

$$C = \begin{bmatrix} \mathbf{2} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array}$$

Matriz de requisições

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow \mathbf{P_1} \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array}$$

Deadlocks

Ao final da execução, temos:

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

Recursos existentes

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

Recursos disponíveis

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

Matriz de alocação

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

Matriz de requisições

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

Deadlocks – Situação 1

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

Requisições:

P_2 requisita duas unidade de fita, um *scanner*
e uma unidade de CD-ROM;

Recursos disponíveis

Recursos existentes

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

$A = (2 \ 1 \ 0 \ 0)$ P_3 pode executar

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

Matriz de alocação

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array}$$

Matriz de requisições

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \longleftarrow P_1 \\ \longleftarrow P_2 \\ \longleftarrow P_3 \end{array}$$

Nessa situação, nenhum processo pode ser atendido!

DEADLOCK

Deadlocks

- Detecção com vários recursos de cada tipo:
 - Para esse algoritmo, o sistema, geralmente, procura periodicamente por *deadlocks*;
 - CUIDADO:
 - Evitar ociosidade da CPU → quando se tem muitos processos em situação de *deadlock*, poucos processos estão em execução;

Deadlocks

- Recuperação de *Deadlocks*:
 - Por meio de preempção: possibilidade de retirar temporariamente um recurso de seu atual dono (processo) e entregá-lo a outro processo;
 - Por meio de *rollback*: recursos alocados a um processo são armazenados em arquivos de verificação; quando ocorre um *deadlock*, os processos voltam ao estado no qual estavam antes do *deadlock* → **solução cara**;

Deadlocks

- Recuperação de *Deadlocks*:
 - Por meio de eliminação de processos: processos que estão no ciclo com *deadlock* são retirados do ciclo;
 - Melhor solução para processos que não causam algum efeito negativo ao sistema;
 - Ex1.: compilação – sem problemas;
 - Ex2.: atualização de um base de dados – problemas;

Deadlocks

- **(3) Evitar dinamicamente o problema:**
 - Alocação individual de recursos → à medida que o processo necessita;
 - Soluções também utilizam matrizes;
 - 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;
 - Definição de Estados Seguros e Inseguros;

Deadlocks

- 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;

Deadlocks

- Algoritmos do Banqueiro:
 - Idealizado por Dijkstra (1965);
 - Considera cada requisição no momento em que ela ocorre, verificando se essa requisição leva a um estado seguro; Se sim, a requisição é atendida, se não o atendimento é adiado para um outro momento;
 - Premissas adotadas por um banqueiro (SO) para garantir ou não crédito (recursos) para seus clientes (processos);
 - Nem todos os clientes (processos) precisam de toda a linha de crédito (recursos) disponível para eles;

Deadlocks

- Algoritmo do Banqueiro para um único tipo de recurso:

Possui

Máximo de linha de crédito = 22

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

Livre: 10

Seguro

A	1	6
B	1	5
C*	2	4
D	4	7

Livre: 2

Seguro

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

Livre: 1

Inseguro

- Solicitações de crédito são realizadas de tempo em tempo;
- * C é atendido e libera 4 créditos, que podem ser usados por B ou D;

Deadlocks

- Algoritmo do Banqueiro para um único tipo de recurso:

Possui

Máximo de linha de crédito = 22

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

Livre: 10

Seguro

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

Livre: 2

Seguro

A	1	6
B*	2	5
C	2	4
D	4	7

Livre: 1

Inseguro

- Solicitações de crédito são realizadas de tempo em tempo;
- * B é atendido. Em seguida os outros fazem solicitação, ninguém poderia ser atendido;

Deadlocks

- Algoritmo do Banqueiro para vários tipos de recursos:
 - Mesma ideia, mas duas matrizes são utilizadas;

	Processos	Unidade de Fita	Plotters	Impressoras	Unidade 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	

C = Recursos Alocados

Recursos $\rightarrow E = (6 \ 3 \ 4 \ 2)$;
 Alocados $\rightarrow P = (5 \ 3 \ 2 \ 2)$;
 Disponíveis $\rightarrow A = (1 \ 0 \ 2 \ 0)$;

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

R = Recursos ainda necessários

Deadlocks

- Algoritmo do Banqueiro para vários tipos de recursos:

	Processos	Unidade de Fita	Plotters	Impressoras	Unidade de CD-ROM
A	3	0	0	1	1
B	0	1	1	1	0
C	1	1	1	1	0
D	1	1	1	0	1
E	0	0	0	0	0

Alocados $\rightarrow P = (5 \ 3 \ \mathbf{3} \ 2)$;
 Disponíveis $\rightarrow A = (1 \ 0 \ \mathbf{1} \ 0)$;

A	1	1	0	0
B	0	1	0	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

C = Recursos Alocados R = Recursos ainda necessários

- Podem ser atendidos: D, A ou E, C;

Deadlocks

- Algoritmo do Banqueiro para vários tipos de recursos:

	Processos	Unidade de Fita	Plotters	Impressoras	Unidade de CD-ROM
A	3	0	1	1	
B	0	1	1	0	
C	1	1	1	0	
D	1	1	0	1	
E	0	0	1	0	

C = Recursos Alocados

Alocados $\rightarrow P = (5 \ 3 \ 4 \ 2)$;

Disponíveis $\rightarrow A = (1 \ 0 \ 0 \ 0)$;

A	1	1	0	0
B	0	1	0	2
C	3	1	0	0
D	0	0	1	0
E	2	1	0	0

R = Recursos ainda necessários

- *Deadlock* \rightarrow atender o processo E;
- Solução: Adiar a requisição de E por alguns instantes;

Deadlocks

- Algoritmo do Banqueiro:
 - Desvantagens
 - Pouco utilizado, pois é difícil saber quais recursos serão necessários;
 - Escalonamento cuidadoso é caro para o sistema;
 - O número de processos é dinâmico e pode variar constantemente, tornando o algoritmo custoso;
 - Vantagem
 - Na teoria o algoritmo é ótimo;

Deadlocks

- **(4) Prevenir Deadlocks:**

- Atacar uma das quatro condições:

Condição	Abordagem
-----------------	------------------

Exclusão Mútua	<p>Não é viável deixar de usar;</p> <p>Alocar todos os recursos usando um <i>spool</i> - possível para alguns recursos, por ex. impressora.</p> <p>Só alocar recurso quando for absolutamente necessário.</p>
Uso e Espera	<p>Requisitar todos os recursos inicialmente - processo não sabe o que será necessário (volta ao algoritmo do banqueiro)</p> <p>Uso inadequado dos recursos</p> <p>Quando processo quer mais um recurso deve liberar todos os que possui e requisitar todos novamente</p>

Deadlocks

- Atacar uma das quatro condições:

Condição	Abordagem
Não-preempção	Retirar recursos dos processos Pode ocasionar problemas de desempenho - fita Pode ocasionar erros - CDROM Opção pouco aconselhável!
Espera Circular	Ordenar numericamente os recursos Requisitar recursos só em ordem crescente Pode ser ineficiente mas é a mais atrativa de ser praticada

Deadlocks

- Problema potencial em qualquer SO
- Pode-se utilizar:
 - Algoritmos para prevenir deadlock
 - Algoritmos para detectar deadlock
- Algoritmos causam impacto negativo no desempenho e flexibilidade do sistema
- SOs de propósito geral utilizam Algoritmo do Avestruz