

SSC 140 - SISTEMAS OPERACIONAIS I

Turmas A e B

Aula 11 – Deadlocks

Profa. Sarita Mazzini Bruschi

Slides de autoria de
Luciana A. F. Martimiano 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;

2

Deadlocks



Uma situação de *deadlock*

3

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;

4

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;

5

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 ao um deadlock ou não

6

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."

7

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*

8

Deadlocks

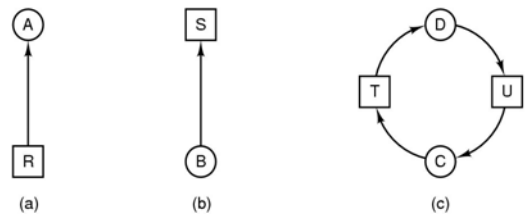
- Geralmente, *deadlocks* são representados por grafos a fim de facilitar sua detecção, prevenção e recuperação

- Ocorrência de ciclos pode levar a um *deadlock*;

9

Deadlocks

Grafos de alocação de recursos



- a) Recurso R alocado ao Processo A
- b) Processo B requisita Recurso S
- c) **Deadlock**

10

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;

11

Deadlocks

(1) Ignorar o problema:

- Frequência do problema;
- Alto custo – estabelecimento de condições para o uso de recursos;
- UNIX e WINDOWS;
- Algoritmo do AVESTRUZ;

12

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;

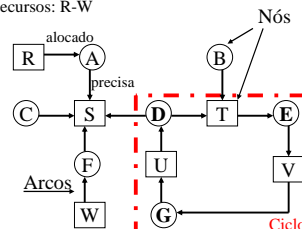
13

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*?

14

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;

15

Deadlocks

4 unidades de fita;
2 *plotter*;
3 *scanners*;
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 dois *scanners*;
Cada processo precisa de outros recursos (R);

Recursos existentes

$E = (4 \ 2 \ 3 \ 1)$
UF P S UCD

Recursos disponíveis

$A = (2 \ 1 \ 0 \ 0)$
UF P S UCD

Matriz de alocação

$C = \begin{bmatrix} \text{UF} & \text{P} & \text{S} & \text{UCD} \\ 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$ ← P_1
← P_2
← P_3

Matriz de requisições

$R = \begin{bmatrix} \text{UF} & \text{P} & \text{S} & \text{UCD} \\ 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$ ← P_1
← P_2
← P_3

Recursos

16

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

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

Recursos disponíveis

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

Matriz de alocação

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

Matriz de requisições

$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ ← P_1
← P_2
← $P_{3,7}$

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

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

Recursos disponíveis

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

Matriz de alocação

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

Matriz de requisições

$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ ← P_1
← P_2
← $P_{3,18}$

Deadlocks

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

Requisições:

P₁ requisita duas unidades de fita e um CD-ROM;
P₂ requisita uma unidade de fita e um *scanner*;
P₃ requisita duas unidades de fita e um *plotter*;

Recursos existentes

E = (4 2 3 1)

Recursos disponíveis

A = (2 1 0 0)

A = (2 2 2 0) **P₂ pode executar**

A = (1 2 1 0)

Matriz de alocação

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

Matriz de requisições

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

Deadlocks

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

Requisições:

P₁ requisita duas unidades de fita e um CD-ROM;
P₂ requisita uma unidade de fita e um *scanner*;
P₃ requisita duas unidades de fita e um *plotter*;

Recursos existentes

E = (4 2 3 1)

Recursos disponíveis

A = (2 1 0 0)

A = (2 2 2 0) **P₂ pode executar**

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

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

Deadlocks

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

Requisições:

P₁ requisita duas unidades de fita e um CD-ROM;
P₂ requisita uma unidade de fita e um *scanner*;
P₃ requisita duas unidades de fita e um *plotter*;

Recursos existentes

E = (4 2 3 1)

Recursos disponíveis

A = (2 1 0 0)

A = (2 2 2 0)

A = (2 2 1 0) **P₁ pode executar**

Matriz de alocação

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

Matriz de requisições

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

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

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

Matriz de requisições

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

Deadlocks – Situação 1

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

Requisições:

P₂ requisita duas unidades de fita, um *scanner* e uma unidade de CD-ROM;

Recursos existentes

E = (4 2 3 1)

Recursos disponíveis

A = (2 1 0 0) **P₃ pode executar**

A = (2 2 2 0)

Matriz de alocação

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

Matriz de requisições

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

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**;

25

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;

26

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;

27

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;

28

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;

29

Deadlocks

Algoritmo do Banqueiro para um único tipo de recurso:

Possui	Máximo de linha de crédito = 22																																					
<table border="1"> <tr><td>A</td><td>0</td><td>6</td></tr> <tr><td>B</td><td>0</td><td>5</td></tr> <tr><td>C</td><td>0</td><td>4</td></tr> <tr><td>D</td><td>0</td><td>7</td></tr> </table>	A	0	6	B	0	5	C	0	4	D	0	7	<table border="1"> <tr><td>A</td><td>1</td><td>6</td></tr> <tr><td>B</td><td>1</td><td>5</td></tr> <tr><td>C*</td><td>2</td><td>4</td></tr> <tr><td>D</td><td>4</td><td>7</td></tr> </table>	A	1	6	B	1	5	C*	2	4	D	4	7	<table border="1"> <tr><td>A</td><td>1</td><td>6</td></tr> <tr><td>B</td><td>2</td><td>5</td></tr> <tr><td>C</td><td>2</td><td>4</td></tr> <tr><td>D</td><td>4</td><td>7</td></tr> </table>	A	1	6	B	2	5	C	2	4	D	4	7
A	0	6																																				
B	0	5																																				
C	0	4																																				
D	0	7																																				
A	1	6																																				
B	1	5																																				
C*	2	4																																				
D	4	7																																				
A	1	6																																				
B	2	5																																				
C	2	4																																				
D	4	7																																				
Livre: 10	Livre: 2	Livre: 1																																				
Seguro	Seguro	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;

30

Deadlocks

- Algoritmo do Banqueiro para um único tipo de recurso:

Possui

Máximo de linha de crédito = 22

	A	B	C	D
Processos	0	0	0	0
Unidade de Fila	6	5	4	7
Ploters	1	1	2	4
Impressoras	1	1	2	4
Unidade de CD-ROM	6	5	4	7

Livre: 10 Livre: 2 Livre: 1

Seguro **Seguro** **Inseguro**

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

31

Deadlocks

- Algoritmo do Banqueiro para vários tipos de recursos:

- Mesma idéia, mas duas matrizes são utilizadas;

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

	A	B	C	D	E
Processos	3	0	1	1	0
Unidade de Fila	1	1	0	0	0
Ploters	1	1	1	0	0
Impressoras	1	1	0	1	0
Unidade de CD-ROM	1	1	0	1	0

C = Recursos Alocados R = Recursos ainda necessários

32

Deadlocks

- Algoritmo do Banqueiro para vários tipos de recursos:

Alocados → P = (5 3 3 2);
Disponíveis → A = (1 0 1 0);

	A	B	C	D	E
Processos	3	0	1	1	0
Unidade de Fila	1	1	1	0	1
Ploters	1	1	1	0	1
Impressoras	1	1	0	1	0
Unidade de CD-ROM	1	1	0	1	0

C = Recursos Alocados R = Recursos ainda necessários

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

33

Deadlocks

- Algoritmo do Banqueiro para vários tipos de recursos:

Alocados → P = (5 3 4 2);
Disponíveis → A = (1 0 0 0);

	A	B	C	D	E
Processos	3	0	1	1	0
Unidade de Fila	1	1	1	0	1
Ploters	1	1	1	0	1
Impressoras	1	1	0	1	0
Unidade de CD-ROM	1	1	0	1	0

C = Recursos Alocados R = Recursos ainda necessários

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

34

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;

35

Deadlocks

- Prevenir Deadlocks:

- Atacar uma das quatro condições:

Condição	Abordagem
Exclusão Mútua	Alocar todos os recursos usando um <i>spool</i>
Posse e Espera	Requisitar todos os recursos inicialmente para execução – difícil saber; sobrecarga do sistema
Não-preempção	Retirar recursos dos processos – pode ser ruim dependendo do tipo de recurso; praticamente não implementável
Espera Circular	Ordenar numericamente os recursos, e realizar solicitações em ordem numérica Permitir que o processo utilize apenas um recurso por vez

36

Deadlocks

- Deadlocks podem ocorrer sem o envolvimento de recursos, por exemplo, se semáforos forem implementados erroneamente:

```
..          ..
down(&empty); down(&mutex);
down(&mutex); down(&empty);
..          ..
```

- Inanição (*Starvation*)

- Todos os processos devem conseguir utilizar os recursos que precisam, sem ter que esperar indefinidamente;
- Alocação usando FIFO;

37

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);
}
```

38