FATEC

CENTRO PAULA SOUZA
COMPETÊNCIA EM EDUCAÇÃO PÚBLICA PROFISSIONAL
Faculdade de Tecnologia Prof. Jessen Vidal
São José dos Campos

GOVERNO DO ESTADO DE SÃO PAULO

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Disciplina: Sistemas Operacionais I

Aula 14: Deadlocks P1

Prof. Diogo Branquinho Ramos
diogo.branquinho@fatec.sp.gov.br
São José dos Campos - SP

Roteiro

- O problema de deadlock
- Características do sistema computacional
- Caracterização do deadlock
  - As quatro condições
- Grafo de alocação
- Métodos para tratamento de deadlocks
  - Prevenção de deadlock

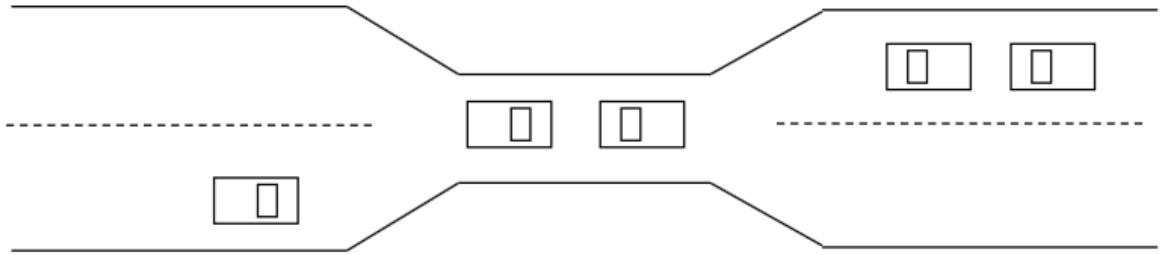
O problema do deadlock

- **Um conjunto de processos bloqueados, cada um mantendo um recurso e esperando para adquirir um recurso mantido por outro processo no conjunto.**
- **Exemplo**
  - Sejam S e Q dois semáforos inicializados com 1

P_0	P_1
S.acquire();	Q.acquire();
Q.acquire();	S.acquire();
.	.
S.release();	Q.release();
Q.release();	S.release();

Metáfora do cruzamento da ponte

- Tráfego apenas em uma direção (recurso limitado).
- Cada seção de uma ponte pode ser vista como um recurso.
- Se houver deadlock, ele pode ser resolvido se um carro parar (ceder o recurso).
- Vários carros podem ter que parar se houver um deadlock.
- É possível haver starvation.



Características do sistema computacional

- **Limitação**
  - Quantidade finita de recursos a serem distribuídos entre vários processos em competição.
- **Recursos podem ser classificados em tipos**
  - Ciclos de CPU, espaço de memória, dispositivos de E/S...
  - Cada tipo de recurso R_i possui instâncias W_i.
- **Recurso preemptível**
  - Aquele que pode ser tomado do processo sem prejuízo.
    - Registrador.
- **Recurso não preemptível**
  - Aquele que não pode ser tomado do processo.
    - Impressora.

Características do sistema computacional

- **Cada processo utiliza um recurso da seguinte forma:**
  - Requisição: solicitação do recurso.
    - Se estiver ocupado, o processo poderá esperar.
  - Uso: operação no recurso.
  - Liberação: "devolução" do recurso.

Caracterização do deadlock

Quatro condições simultâneas

- **Exclusão mútua**
  - Apenas um processo de cada vez pode usar um recurso.
- **Manter e esperar**
  - Um processo mantendo pelo menos um recurso está esperando para adquirir outros recursos mantidos por outros processos.
- **Não-preempção**
  - Um recurso só pode ser liberado voluntariamente pelo processo que o mantém, depois que esse processo tiver terminado sua tarefa.

Caracterização do deadlock

Quatro condições simultâneas

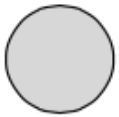
- **Espera circular**
  - Existe um conjunto $\{P_0, P_1, \ldots, P_0\}$ de processos esperando tal que P_0 está esperando por um recurso que é mantido por P_1, P_1 está esperando por um recurso que é mantido por P_2, ..., P_{n-1} está esperando por um recurso que é mantido por P_n, e P_n está esperando por um recurso que é mantido por P_0.

Grafo de alocação de recursos

- **Um conjunto de vértices V e um conjunto de arestas A.**
- **V é particionado em dois tipos:**
  - $P = \{P_1, P_2, \ldots, P_n\}$
    - Conjunto consistindo em todos os processos no sistema.
  - $R = \{R_1, R_2, \ldots, R_m\}$
    - Conjunto consistindo em todos os tipos de recurso no sistema.
- **Aresta de requisição – aresta direcionada $P_i \rightarrow R_j$**
- **Aresta de atribuição – aresta direcionada $R_j \rightarrow P_i$**

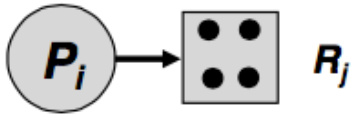
Grafo de alocação de recursos

- **Processo**

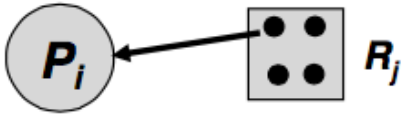


- **Tipo de recurso com 4 instâncias**

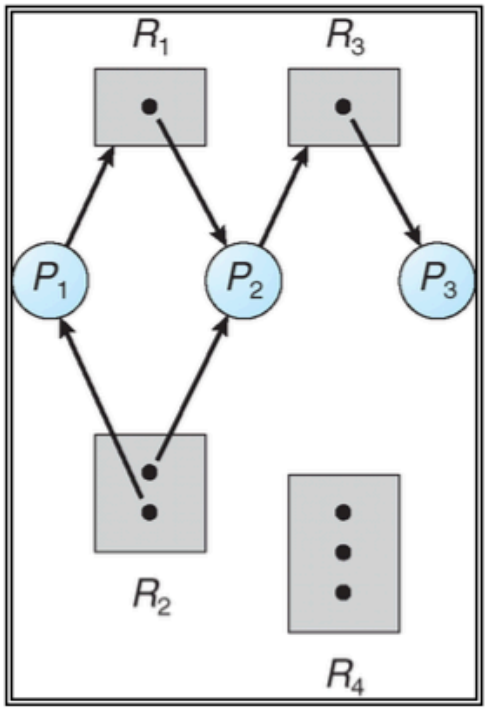
- **P_i solicita instância de R_j**



- **P_i está mantendo uma instância de R_j**



Exemplo de grafo

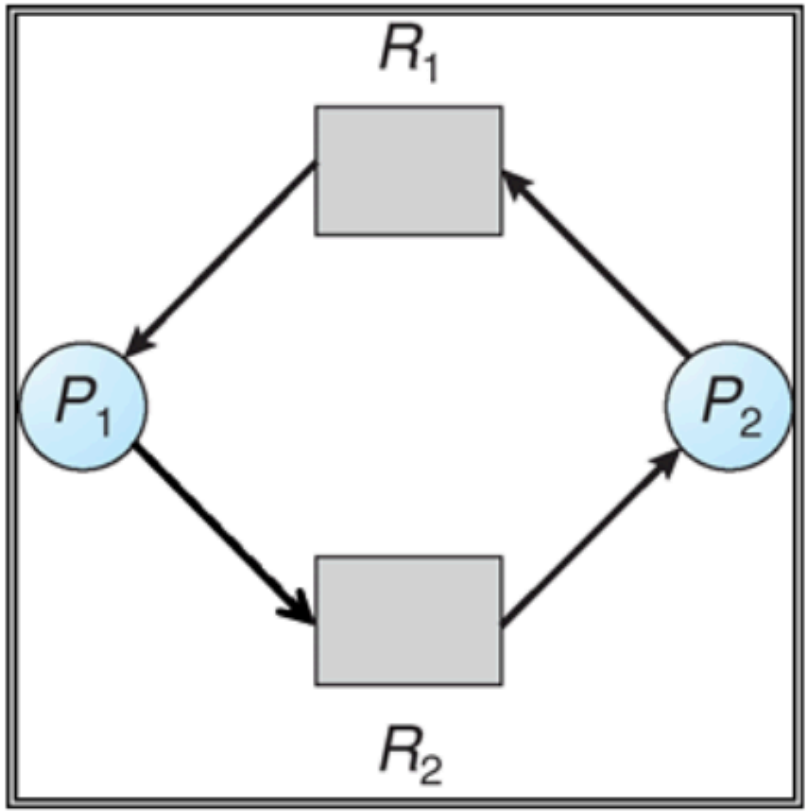


- **Conjuntos P, R e A**
  - $P=\{P_1,P_2,P_3\}$
  - $R=\{R_1,R_2,R_3,R_4\}$
  - $A=\{P_1 \rightarrow R_1,\ P_2 \rightarrow R_3,\ R_1 \rightarrow P_2,\ R_2 \rightarrow P_2,\ R_2 \rightarrow P_1,\ R_3 \rightarrow P_3\}$
- **Instâncias**
  - 1 instância de R_1
  - 2 instâncias de R_2
  - 1 instância de R_3
  - 3 instâncias de R_4
- **Estados**

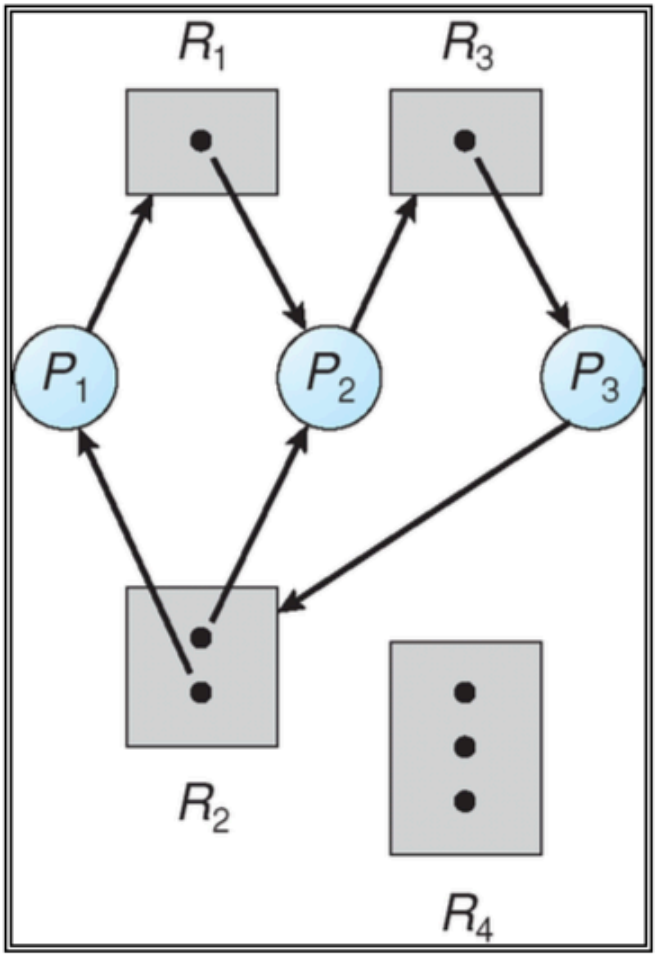
Algumas definições

- **Se o grafo não contém ciclos**
  - Nenhum processo está em deadlock.
- **Se cada recurso possuir apenas uma instância**
  - Um ciclo indica deadlock em todos os processos do ciclo.
- **Se cada recurso possuir várias instâncias**
  - Um ciclo não indica necessariamente um deadlock.

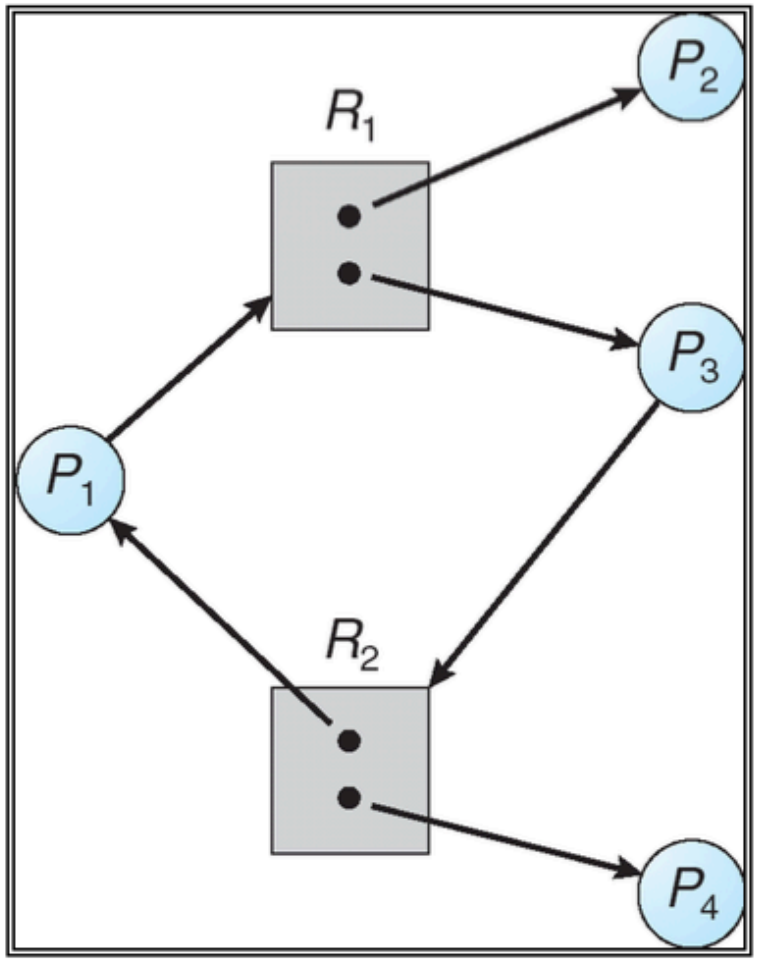
Exemplo de grafo com deadlock



Exemplo de grafo com deadlock



Exemplo de grafo com ciclo sem deadlock



Resumo

- **Se o grafo não contém ciclos**
  - Sem deadlock!
- **Se o grafo contém um ciclo**
  - Se há apenas uma instância por tipo de recurso, então existe o deadlock;
  - Se há várias instâncias por tipo de recurso, possibilidade de deadlock.

Métodos para tratamento de deadlocks

- **Garantir que o sistema nunca entrará em um estado de deadlock.**
  - Prevenir: quebrar uma das quatro condições.
  - Evitar: gerenciar o atendimento de requisições.
- **Permitir que o sistema entre em um estado de *deadlock* e depois se recupere.**
- **Algoritmo da Avestruz**
  - Ignorar o problema: engenheiros atentos às estatísticas.
  - Maioria dos SOs: UNIX e Windows. A JVM também.
  - O sistema está em deadlock, mas não reconhece isto.
  - Degradação até ser reiniciado manualmente.

Prevenção de deadlock

- **Exclusão mútua**
  - Não exigida para recursos compartilháveis (leitura de um arquivo), mas deve ser mantida para recursos não compartilháveis (impressora) entre processos.
- **Manter e esperar (processo libera)**
  - Deve garantir que sempre que um processo solicita um recurso, ele não mantém quaisquer outros recursos.
    - Exige que o processo solicite e tenha todos os seus recursos alocados antes de iniciar a execução, ou permite que o processo solicite recursos somente quando o processo não tiver utilizando outros recursos.
      - Ex: cópia de dados de um DVD para o HD e impressão.
    - Problemas: baixa utilização de recursos; starvation possível.

Prevenção de deadlock

- **Sem preempção (recurso é tomado)**
  - Se um processo que está mantendo alguns recursos solicitar outro recurso que não pode ser alocado imediatamente a ele, então todos os recursos atualmente mantidos são preemptados.
  - Recursos preemptados são acrescentados à lista de recursos pelos quais o processo está esperando.
  - O processo só será reiniciado quando puder reaver seus antigos recursos, além dos novos que está solicitando.

Prevenção de deadlock

- **Sem preempção**
  - P requisita recursos: se estiverem disponíveis, alocamos;
  - Se os recursos estão com Q e Q estiver esperando por outros recursos, preemptamos os recursos de Q, para que P os use;
  - Se Q não estiver esperando, os recursos de P são preemptados e P espera!

Prevenção de deadlock

- **Espera circular**
  - Impõe uma ordenação total de todos os tipos de recursos, e exige que cada processo solicite recursos em uma ordem de enumeração aumentada.
  - Ex:
    - F(fita) = 1
    - F(disco) = 5
    - F(impressora) = 12

→Um processo que desejar usar a fita e a impressora terá de requisitar primeiro a fita e depois a impressora.

→Paga-se o preço do starvation para não ter deadlock!

Prevenção de deadlock

- **Espera circular**
  - Prova por contradição
    - Seja o conjunto de processos envolvidos na espera circular $\{P_0, P_1,...,P_n\}$
    - P_i espera pelo recurso R_i mantido por P_{i+1}: para isso, enquanto requisitarmos R_{i+1}, precisamos ter $F(R_i)<F(R_{i+1})$, para todo i.
    - Essa condição significa que $F(R_0)<F(R_1)<...<F(R_n)<F(R_0)$.
    - Pela transitividade, $F(R_0)<F(R_0)$, o que é impossível!
    - Portanto, não pode haver espera circular.