Aula 12: Deadlock

Deadlock (interbloqueio, blocagem, impasse) refere-se a uma situação em que ocorre um **impasse**, e dois ou mais processos ficam **impedidos** de continuar suas execuções - ou seja, ficam bloqueados, esperando uns pelos outros.

O deadlock ocorre com um conjunto de processos e recursos não-preemptíveis, onde um ou mais processos desse conjunto está **aguardando** a liberação de um recurso por um outro processo, o qual, por sua vez, aguarda a liberação de outro recurso alocado ou dependente do primeiro processo.

A definição textual de deadlock por ser muito abstrata, é mais difícil de se compreender do que a representação por grafos, que será resumida mais adiante. No entanto, algumas observações são pertinentes:

- O deadlock pode ocorrer mesmo que haja somente um processo no sistema operacional, considerando que este processo utilize múltiplos threads e que tais threads requisitem os recursos alocados a outros threads no mesmo processo;
- O deadlock independe da quantidade de recursos disponíveis no sistema;
- Normalmente o deadlock ocorre com recursos, tais como dispositivos, arquivos, memória etc. Apesar de a CPU também ser um recurso para o sistema operacional, em geral é um recurso facilmente preemptível, pois existem os escalonadores para compartilhar o processador entre os diversos processos, quando trata-se de um ambiente multitarefa.

Condições necessárias para a ocorrência de deadlock

O deadlock ocorre naturalmente em alguns sistema operacional. No entanto, é necessário ressaltar que tais sistemas precisam obedecer a algumas condições para que uma situação de deadlock se manifeste.

Essas condições estão listadas abaixo. As três primeiras caracterizam um modelo de sistema, e a última é o deadlock propriamente dito. Processos que estejam de posse de recursos obtidos anteriormente podem solicitar novos recursos. Caso estes recursos já estejam alocados a outros processos, o processo solicitante deve aguardar pela liberação do mesmo:

- **Condição de não-preempção**: recursos já alocados a processos não podem ser tomados à força. Eles precisam ser liberados explicitamente pelo processo que detém a sua posse;
- **Condição de exclusividade mútua**: cada recurso ou está alocado a um processo ou está disponível;

- **Condição de posse-e-espera**: cada processo pode solicitar um recurso, ter esse recurso alocado para si e ficar bloqueado, esperando por um outro recurso;
- **Condição de espera circular**: deve existir uma cadeia circular de dois ou mais processos, cada um dos quais esperando por um recurso que está com o próximo integrante da cadeia.

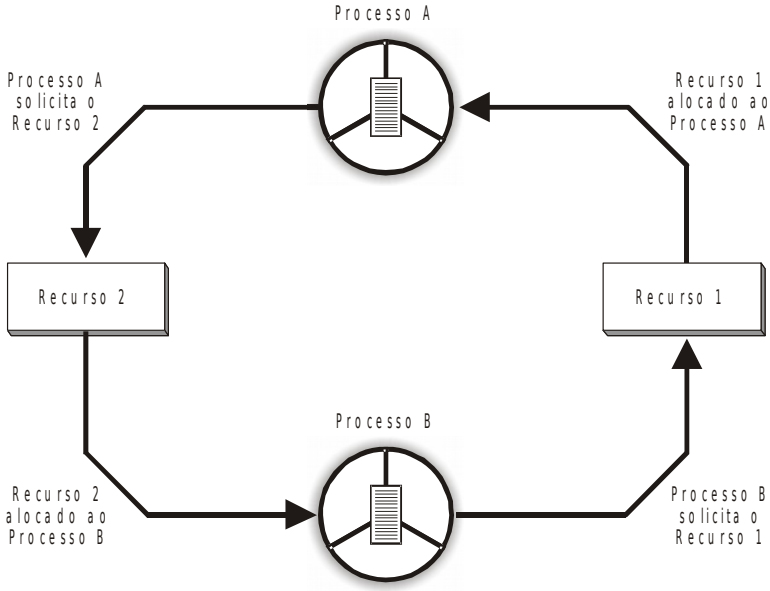


Figura 1: Espera Circular.

Tratamento de deadlock

As situações de deadlock **podem ser tratadas ou não** em um sistema, e cabe aos desenvolvedores avaliar o custo/benefício que essas implementações podem trazer. Normalmente, as estratégias usadas para detectar e tratar as situações de deadlocks geram grande sobrecarga, podendo até causar um dano maior que a própria ocorrência do deadlock, sendo, às vezes, melhor ignorar a situação. Existem três estratégias para tratamento de deadlocks:

- Ignorar a situação;
- Detectar o deadlock e recuperar o sistema;
- Evitar o deadlock.

Ignorar a situação

A estratégia mais simples para tratamento (ou não) do "deadlock", conhecida como **Algoritmo do Avestruz**, é simplesmente ignorá-lo. Muitos defendem que a freqüência de ocorrência deste tipo de evento é baixa demais para que seja necessário sobrecarregar a CPU com códigos extras de tratamento, e que, ocasionalmente, é tolerável reiniciar o sistema como uma ação corretiva.

Prevenção do Deadlock

Para prevenir o Deadlock é preciso garantir que uma das quatro condições acima citada nunca ocorra, dentre as diversas situações já citadas pode ser feito um minucioso trabalho de determinar muito bem que recursos, quais recursos e quando estes recursos deverão ser disponibilizados aos processos.

Detectar o deadlock e recuperar o sistema

Nessa estratégia, o sistema permite que ocorra o deadlock e só então executa o procedimento de recuperação, que resume-se na detecção da ocorrência e na recuperação posterior do sistema. É na execução desse procedimento que ocorre a sobrecarga, pois existem dois grandes problemas: primeiramente, como/quando detectar o deadlock e depois, como corrigi-lo.

Para detectar o deadlock, o sistema deve implementar uma estrutura de dados que armazene as informações sobre os processos e os recursos alocados a eles. Essas estruturas deverão ser atualizadas dinamicamente, de modo que reflitam realmente a situação de cada processo/recurso no sistema.

Só o mero procedimento de atualização dessas estruturas já gera uma sobrecarga no sistema, pois toda vez que um processo aloca, libera ou requisita um recurso, as estruturas precisam ser atualizadas.

Além disso, o sistema operacional precisa verificar a ocorrência da condição de espera circular nessas estruturas para a efetiva detecção do deadlock. Esse procedimento, por sua vez, gera outra sobrecarga, que pode ser mais intensa se não for definido um evento em particular para ser executado, como a liberação de um recurso, por exemplo. Assim, ou o sistema operacional verifica periodicamente as estruturas (o que não é aconselhável, pois pode aumentar consideravelmente o tempo de espera dos processos não-bloqueados), ou pode-se implementar uma política, onde o sistema operacional verifica as estruturas quando o mesmo realizar algum procedimento de manutenção do sistema, por exemplo.

Finalmente, só após detectar a presença do deadlock no sistema, o sistema operacional precisa corrigi-lo, executando um procedimento de recuperação. Quanto à detecção do deadlock, vamos apresentar uma das técnicas usadas para detectar a ocorrência de deadlock em sistemas que possuem vários recursos de cada tipo.

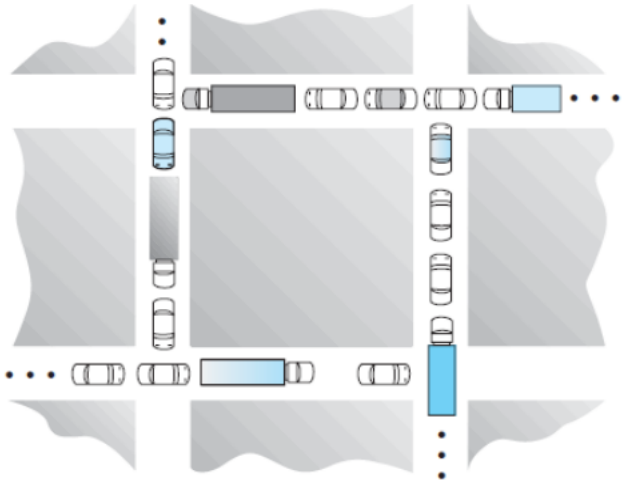
Exercícios

1) Liste três exemplos de deadlocks não relacionados a um ambiente de computação.

2) É possível haver um deadlock envolvendo apenas um processo? Explique sua resposta.

3) Considere um sistema composto de quatro recursos do mesmo tipo compartilhados por três processos, cada um deles necessitando de no máximo dois recursos. Mostre que o sistema é livre de deadlock.

4) Considere o deadlock de tráfego indicado na figura abaixo:



a) Mostre que as quatros condições para o deadlock de fato estão presentes nesse exemplo.

b) Apresente uma regra simples que evite deadlock nesse sistema.