

Deadlocks

Referência:

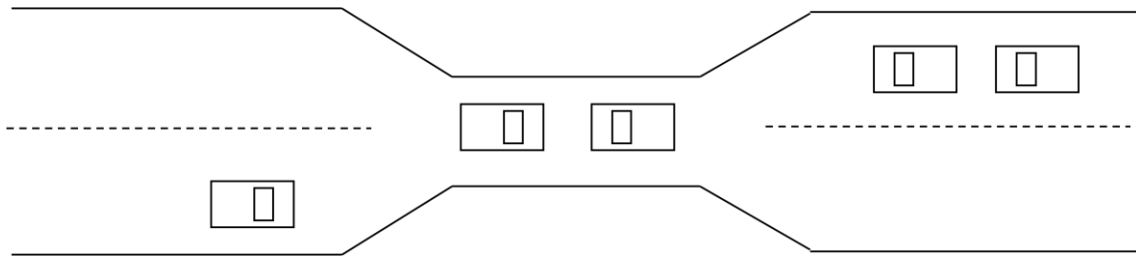
Silberschatz, Abraham. Sistemas Operacionais com Java. 7 ed. Rio de Janeiro: Elsevier, 2008.

- Modelo
- Caracterização do Deadlock
- Métodos para tratamento de Deadlocks
- Prevenção de Deadlocks
- Evitar Deadlock
- Detectar Deadlock
- Recuperação do Deadlock
- Técnica combinada para tratamento de Deadlocks

O Problema do Deadlock

- Um conjunto de processos bloqueados, cada um retendo um recurso e esperando para adquirir um recurso retido de outro processo no conjunto.
- Exemplo
 - O sistema possui 2 unidades de fita.
 - P_1 e P_2 mantêm, cada um, uma unidade de fita, e cada uma delas precisa de outra unidade.
- Exemplo
 - Os semáforos A e B , inicializados como 1.

P_0	P_1
$wait(A);$	$wait(B)$
$wait(B);$	$wait(A)$



- Tráfego em um único sentido.
- Cada seção de uma ponte pode ser vista como um recurso.
- Se ocorrer um deadlock, ele pode ser resolvido se um dos carros recuar (preemptar recursos e reverter).
- Vários carros podem ter de recuar se um deadlock ocorrer.
- É possível haver Starvation.

Modelo do Sistema

- Tipos de recursos $R_1, R_2, \dots, R_m$.

Ciclos de CPU, espaço de memória, dispositivos de E/S

- Cada tipo de recurso R_i possui W_i instâncias.
- Cada processo usa um recurso desta forma:
 - requisitar
 - usar
 - liberar
- Pode haver deadlock se quatro condições ocorrerem simultaneamente
 - **Exclusão mútua:** Somente um processo de cada vez pode usar um recurso.
 - **Manter e esperar:** Um processo retendo pelo menos um recurso está esperando para adquirir recursos adicionais mantidos por outros processos.
 - **Sem preempção:** Um recurso pode ser liberado apenas voluntariamente pelo processo que o mantém, após esse processo ter completado sua tarefa.
 - **Espera circular:** Existe um conjunto $\{P_0, P_1, \dots, P_0\}$ de processos esperando de modo que P_0 está esperando um recurso que é mantido por P_1 , P_1 está esperando um recurso que é mantido por P_2 , ..., P_{n-1} está esperando um recurso que é

mantido por P_n , e P_0 está esperando um recurso que é mantido por P_0 .

Métodos para Tratamento de Deadlocks

- Assegura que o sistema nunca entre em um estado de deadlock.
- Permite que o sistema entre em um estado de deadlock e depois se recupere.
- Ignora o problema e finge que o deadlock nunca ocorre no sistema; usado pela maioria dos sistemas operacionais, inclusive o UNIX.
- Restringir as formas como a requisição pode ser feita.
- Exclusão mútua – não necessário para recursos compartilháveis; precisa ser mantida para recursos não compartilháveis.
- Manter e esperar – precisa garantir que sempre que um processo requisitar um recurso, ele não manterá quaisquer outros recursos.
 - Exige que cada processo requisiite e receba a alocação de todos os seus recursos antes de iniciar sua execução, ou permite que um processo requisiite recursos apenas quando não tiver nenhum outro.
 - Baixa utilização de recursos; possibilidade de starvation.
- **Sem preempção**
 - Se um processo que estiver mantendo alguns recursos requisitar outro recurso que não possa ser alocado imediatamente para ele, então todos os recursos atualmente mantidos são liberados.
 - Os recursos preemptados são acrescentados à lista de recursos pelos quais o processo está esperando.
 - O processo será reiniciado somente quando puder reaver seus recursos antigos, assim como os novos que ele está requisitando.
- **Espera circular** – impõe uma ordenação total de todos os tipos de recursos e exige que cada processo requisiite recursos em uma ordem crescente de enumeração.

Evitar Deadlock

- Exige que o sistema tenha algumas informações a priori adicionais disponíveis.

- O modelo mais simples e útil exige que cada processo declare o número máximo de recursos de cada tipo que possa ser necessário.
- O algoritmo para evitar deadlock examina dinamicamente o estado de alocação de recursos para garantir que a condição de espera circular nunca possa existir.
- O estado de alocação de recursos é definido pelo número de recursos disponíveis e alocados e pelas demandas máximas dos processos.

Estado Seguro

- Quando um processo requisita um recurso disponível, o sistema precisa decidir se a alocação imediata deixará o sistema em um estado seguro.
- O sistema está em um estado seguro apenas se houver uma sequência segura de todos os processos.
- Uma sequência $\langle P_1, P_2, \dots, P_n \rangle$ é segura se, para cada P_i , o recurso requisitar que P_i ainda possa ser satisfeito pelos recursos atualmente disponíveis mais os recursos mantidos por todo o P_j , com $j < i$.
 - Se os recursos que P_i precisar não estiverem imediatamente disponíveis, então P_i poderá esperar até que todo o P_j tenha terminado.
 - Quando P_j tiver terminado, P_i poderá obter os recursos necessários, executar, retornar os recursos alocados e terminar.
 - Quando P_i terminar, P_{i+1} poderá obter seus recursos necessários e assim por diante.
- Se um sistema está no estado seguro $\Rightarrow$ nenhum deadlock.
- Se um sistema está no estado inseguro $\Rightarrow$ possibilidade de deadlock.
- Evitar $\Rightarrow$ assegura que um sistema nunca entrará em um estado inseguro.

Detectar Deadlock

- Permite ao sistema entrar no estado de Deadlock.
- Algoritmo de detecção.

- Esquema de recuperação.

Uso do Algoritmo de Detecção

- Quando é necessário invocar o algoritmo depende de:
 - Com que frequência um deadlock provavelmente ocorrerá.
 - Quantos processos precisarão ser revertidos.
 - um para cada ciclo rompido.
- Se o algoritmo de detecção for chamado arbitrariamente, poderá haver muitos ciclos no grafo de recursos e não poderemos dizer quais dos muitos processos em deadlock “causaram” o deadlock.

Recuperação do Deadlock: Término do Processo

- Abortar todos os processos em deadlock.
- Abortar um processo de cada vez até que o ciclo de deadlock seja eliminado.
- Em que ordem devemos escolher abortar?
 - Prioridade do processo.
 - Por quanto tempo o processo esteve em execução e quanto tempo mais levará para ser concluído.
 - Recursos que o processo usou.
 - Recursos de que o processo precisa para ser concluído.
 - Quantos processos precisarão ser terminados.

Se o processo é interativo ou em batch

Recuperação do Deadlock: Preempção de Recursos

- Seleção de uma vítima – minimiza o custo.
- Rollback – retorna a algum estado seguro, reinicia o processo nesse estado.
- Starvation – o mesmo processo pode ser sempre escolhido como vítima; incluir o número de rollback no fator de custo.

Técnica Combinada para Tratamento de Deadlock

- Combina os três métodos básicos:
 - Prevenir

- Impedir
- Detectar

Permitindo o uso de um enfoque ótimo para cada recurso no sistema

- Particiona recursos em classes hierarquicamente ordenadas.
- Usa a técnica mais apropriada para tratar de deadlocks dentro de cada classe.