Deadlock

Sistemas Operacionais

Deadlock

Introdução

É natural que processos de uma aplicação concorrente compartilhem recursos do sistema, como arquivos, dispositivos e E/S e áreas de memória.

O compartilhamento de recursos entre processos pode ocasionar situações indesejáveis, capazes até de comprometer a execução das aplicações.

Para evitar esse tipo de problema, os processos concorrentes devem ter suas execuções sincronizadas, a partir de mecanismos oferecidos pelo sistema operacional, como o objetivo de garantir o processamento correto dos programas.

Deadlock

Aplicações concorrentes

Aplicações concorrentes muitas vezes necessitam que os processos comuniquem-se entre si.

Esta comunicação pode ser implementada por meio de diversos mecanismos, como variáveis compartilhadas na memória principal ou trocas de mensagens.

Nessa situação, é necessário que os processos concorrentes tenham sua execução sincronizada, através de mecanismos do sistema operacional.

Deadlock

Aplicações concorrentes

Os mecanismos que garantem a comunicação entre os processos concorrentes e o acesso a recursos compartilhados são chamados **mecanismos de sincronização**.

No projeto de sistemas operacionais multiprogramáveis, é fundamental a implementação desses mecanismos, para se garantir a integridade e a confiabilidade na execução de aplicações concorrentes.

Deadlock

Deadlock
Deadlock é a situação em que um processo aguarda por um recurso que nunca estará disponível ou um evento que não ocorrerá.

Essa situação é consequência, na maioria das vezes do compartilhamento de recursos, como dispositivos, arquivos e registros, entre processos concorrentes em que a exclusão mútua é exigida.

Deadlock

Deadlock

Para que ocorra a situação de deadlock, quatro condições são necessárias simultaneamente:

Exclusão mútua: cada recurso só pode estar alocado a um único processo, em um determinado instante;

Espera por recurso: um processo, além dos recursos já alocados, pode estar esperando por outros recursos;

Não-preempção: Um recurso não pode ser liberado de um processo só porque outros processos desejam o mesmo recurso;

Espera circular: um processo pode ter de esperar por um recurso alocado ao outro processo e vice-versa.

Deadlock

Prevenção de deadLock
Para prevenir a ocorrência de deadlocks, é preciso garantir que uma das quatro condições apresentadas, necessárias para sua existência nunca aconteça.

A solução mais conhecida para essa situação é o Algorítimo do Banqueiro (Banker´s Algorithm) proposto por Djikstra (1965). Basicamente, esse algoritmo exige que os processos informem o número máximo de cada tipo de recurso necessário à sua execução. Com essas informações, é possível definir o estado de alocação de um recurso, que é a relação entre um número de recursos alocados e disponíveis e o número máximo de processos que necessitam desses recursos.

Deadlock

O **algoritmo do banqueiro** é executado pelo sistema operacional quando um processo de computação requisita recursos.

O algoritmo impede o impasse, ao negar ou adiar o pedido se ele determinar que aceitar o pedido pode colocar o sistema em um estado inseguro (onde um impasse poderia ocorrer). Quando um novo processo entra em um sistema, ele deve declarar o número máximo de instâncias de cada tipo de recurso que não pode exceder o número total de recursos no sistema.

Deadlock

Recursos
Para o algoritmo do banqueiro trabalhar, ele precisa saber três coisas:

1 - Quanto de cada recurso cada processo poderia solicitar
2 -Quanto de cada recurso cada processo atualmente detém
3- Quanto de cada recurso o sistema tem disponível

Recursos podem ser atribuídos a um processo somente se forem preenchidas as seguintes condições:
1.Pedido <= máximo, senão ajuste a condição de erro definida como o processo ultrapassou a quantidade máxima feita por ele.
2.Pedido <= disponível, senão o processo tem que esperar até que mais recursos estejam disponíveis.

Alguns dos recursos que são controlados em sistemas reais são memória, semáforos e interfaces de acesso.

Deadlock

Detecção do deadlock
Para detectar deadlocks, os sistemas operacionais devem manter estruturas de dados capazes de identificar cada recurso de sistema, o processo que o está alocando e os processos que estão à espera da liberação de recurso.

Toda vez que um recurso é alocado ou liberado por um processo, a estrutura deve ser atualizada. Geralmente, os algorítimos que implementam esse mecanismo verificam a existência da espera circular, percorrendo toda a estrutura, sempre que um processo solicita um recurso e ele não pode ser imediatamente garantido.

Deadlock

Correção do Deadlock
Uma solução menos drástica envolve a liberação de apenas alguns recursos alocados aos processo para outros processos, até que o ciclo de espera termine.

Para que essa solução seja, realmente, eficiente, é necessário que o sistema possa suspender um processo, liberar seus recursos e, após a solução do problema, retornar à execução do processo, sem perder o processamento já realizado. Esse mecanismo é conhecido como **rollback** e (além do **overhead** gerado) e é muito difícil de ser implementado, por ser bastante dependente da aplicação que está sendo processada.

Deadlock

Correção do Deadlock
Após a detecção de deadlock, o sistema operacional deverá, de alguma forma, corrigir o problema. Uma solução bastante utilizada pela maioria dos sistemas é eliminar um ou mais processos envolvidos no deadlock e desalocar os recursos já garantidos por eles, quebrando, assim, a espera circular.

A escolha do processo a ser eliminado é feita, normalmente, de forma aleatória ou, então, com base em algum tipo de prioridade. Esse esquema, no entanto, pode consumir considerável tempo do processador, gerando elevado **overhead** ao sistema.