Sistemas Operacionais
Comunicação de Processos 3

Prof. Alexandre Beletti

O Problema dos Leitores e Escritores

- Proposto por Courtois (1971)
- Modela o acesso a um banco de dados
- Ex: Sistema de reserva de passagens de uma companhia aérea
- Muitos processos concorrentes querendo ler e escrever
- Permite-se que múltiplos processos leiam o banco de dados, mas se algum processo estiver atualizando (escrevendo), nenhum outro processo poderá ter acesso ao banco de dados.

O Problema dos Leitores e Escritores

- O primeiro leitor que recebe acesso ao BD faz um **DOWN** no semáforo.
- Os próximos leitores meramente incrementam um contador (*rc*).
- A medida que os leitores saem, eles decrementam o contador (*rc*) e o último faz um **UP** no semáforo, permitindo que um escritor bloqueado, se houve um, entre.

- Imagem Figura 2-19

O Problema dos Leitores e Escritores

- Existe um pequeno "detalhe": quando um **leitor** está lendo e chega outro, nada o impede de ler também o arquivo.
- Porém surge um **escritor**, que não pode ser admitido no banco de dados, uma vez que escritores devem ter acesso exclusivo
- Leitores vão chegando e o escritor fica aguardando, porém ele só executará quando não houver mais nenhum leitor.
- Uma solução: Quando um leitor chega e um escritor está esperando, o leitor aguarda atrás do escritor, ficando suspenso. Obtém um menor paralelismo e um desempenho inferior.

Problema do Barbeiro Adormecido

- 1 barbeiro, 1 cadeira de barbeiro e *n* cadeiras para os clientes esperarem
- Se não houver nenhum cliente o barbeiro senta na cadeira e dorme
- Quando um cliente chega ele acorda o barbeiro
- Se outros clientes chegarem e o barbeiro estiver cortando, eles verificam se existem cadeiras vazias:
  - Se existem, eles aguardam
  - Se não existem, eles saem

Problema do Barbeiro Adormecido

- Existem três semáforos:
  - Customers: conta os clientes que esperam
  - Barbers: barbeiros desocupados (0 ou 1)
  - Mutex: utilizado para exclusão mútua
  - Waiting: também conta os clientes esperando

Problema do Barbeiro Adormecido

Cliente_chega0()
Se numero_de_clientes < numero_cadeiras
  Ficar()
Senao
  Sair()

Problema do Barbeiro Adormecido

Barbeiro_chega()
Barber() //bloquear no semáforo *customers*
        //até alguém chegar

Cliente_chega1()
Customer() //adquire mutex p/ região crítica
Se mutex() = 1
  Cliente_chega0()

Problema do Barbeiro Adormecido

Ficar()
  waiting++
  UP(customers) //acorda o barbeiro

Sair()
  Quit ou Exit

Deadlock

- Dead (morta) + lock (chave, fechadura)
- Processo esperando por algo que nunca vai ocorrer, situação geralmente acarretada por um recurso acessado por diferentes processos e que fazem uso do mesmo através da exclusão mútua.

Deadlock

- Os processos são representado pelas variáveis ‘a’ e ‘b’, desejam fazer uso do mesmo recurso, representado pela função ‘recurso’, que retorna 1 para recurso em uso ou 0 para recurso disponível.
- Quando ‘a’ faz uso do recurso, ele não libera mais o mesmo então o processo ‘b’ fica aguardando e entra em um tipo de “looping” aguardando liberar o recurso.

Deadlock

- Processo faz uso de um recurso por um período indeterminado
- Outro(s) processo(s) fica(m) aguardando.
- Condições de Deadlock (segundo Coffman):
  - "Condição de exclusão mútua. Todo recurso está atribuído a exatamente um processo ou está disponível". – Visto em aulas anteriores.
  - "Condição de segura e espera. Os processos que estão segurando recursos concedidos anteriormente podem solicitar novos recursos". – Próximo slide

Deadlock

- Segura e Espera: Um processo que precisa de N recursos, só poderá acessar qualquer um desses N recursos quando todos o N recursos estiverem disponíveis, caso contrário terá de aguardar (estado de espera).
- **Pode gerar starvation: recursos para a execução nunca estarem todos disponíveis.**

Deadlock

- "Condição de Nenhuma preempção: os recursos previamente concedidos não podem ser tirados de um processo. Eles devem ser explicitamente liberados pelo processo que os está segurando."
- **Para evitar essa condição, o recurso utilizado por um processo é cedido para outro processo que necessita deste mesmo recurso. Assim como o mecanismo para evitar a condição anterior, pode também gerar problemas, como a perca de todo o processamento feito até então pelo processo ou até mesmo *starvation*.**

Deadlock

- "Condição de espera circular: Deve haver uma cadeia circular de dois ou mais processos, cada um dos quais está esperando um recurso segurado pelo próximo membro da cadeia."
- **Uma forma para eliminar essa condição é impedir que um processo possa alocar mais de um recurso de cada vez, a não ser que ele libere o recurso que está fazendo uso no momento.**

Deadlock

- Caso não seja possível evitar a situação de deadlock, deve-se procurar criar estruturas no sistema operacional para monitorar os recursos utilizados pelos processos, tentando prever tal situação.
- Uma outra situação para eliminar o deadlock, que deve ser tratada com muito cuidado, é a eliminação do processo que está retendo o recurso.

Bibliografia

- TANENBAUM, A.S., WOODHULL, A.S., **Sistemas Operacionais Modernos – Projeto e Implementação**. Bookman, 2000.