TÓPICO 3

Deadlocks

Introdução

- Em ambiente monoprogramado geralmente não existe problema: o único processo ativo aloca os recursos dos quais necessita e faz o seu trabalho.
- Em ambiente multiprogramado, existem vários processos competindo por recursos.
  - Ex: impressoras, fitas, etc
- Alguns recursos do computador devem ser alocados exclusivamente a um processo durante o seu uso.

Introdução

- Situação clássica:

```
Processo P1:
Obtem(Ra);
Obtem(Rb);
---processa---
```

```
Processo P2:
Obtem(Rb);
Obtem(Ra);
---processa---
```

Quando os recursos só podem ser usados por um processo de cada vez, os processos P1 e P2 podem chegar a uma situação de bloqueio eterno, esperando por recursos que nunca serão liberados (DEADLOCK).

Recursos

- Recursos são “objetos” que um processo pode adquirir de maneira exclusiva ou não. Na obtenção de recursos de maneira exclusiva, cada recurso só estar alocado a um único processo em um determinado instante.

- O uso de recursos envolve geralmente a seguinte sequência de operações:
  - requisitar -> usar -> liberar

Tipos de Recursos

- *Preemptíveis*: podem ser retirados do processo por uma entidade externa(SO, SGBD, etc). Ex: memória, processador
- *Não-preemptíveis*: não podem ser retirados do processo. O processo que os possui, libera-os de livre e espontânea vontade.

➔Os deadlocks ocorrem normalmente com recursos não preemptíveis.

Definição de Deadlock

- *Um conjunto de processos está em deadlock se cada processo pertencente ao conjunto estiver esperando por um evento que somente um outro processo pertencente ao mesmo conjunto pode fazer ocorrer”*
- Como todos os processos estão esperando, nenhum pode provocar a ocorrência de eventos. Alocação e liberação de recursos são exemplos clássicos de eventos.

Condições Necessárias à Ocorrência de Deadlocks

1. Exclusão mútua: cada recurso deve ser alocado de maneira exclusiva
2. Posse e espera: um processo que detem recursos pode solicitar novos recursos.
3. Não-preempção: cada recurso só pode ser liberado pelo processo que o detem.
4. Espera circular: deve existir um ciclo de processos onde cada processo está esperando por um recurso detido pelo próximo membro da cadeia.

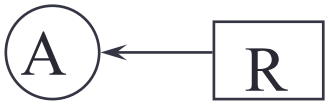
➔Todas as quatro condições devem ocorrer para que haja uma situação de deadlock

Modelo de Deadlock

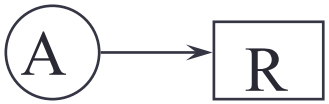
- [Holt 72] propôs o uso de grafos para modelar as situações de deadlock (grafos de alocação de recursos)
- Notação:
  - processos: 

  - recursos: 

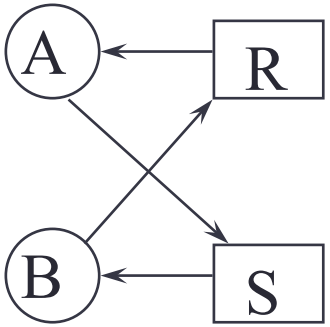
Modelo de Deadlock



: o recurso R está alocado ao processo A



: o processo A deseja obter o recurso R e está bloqueado esperando a sua liberação

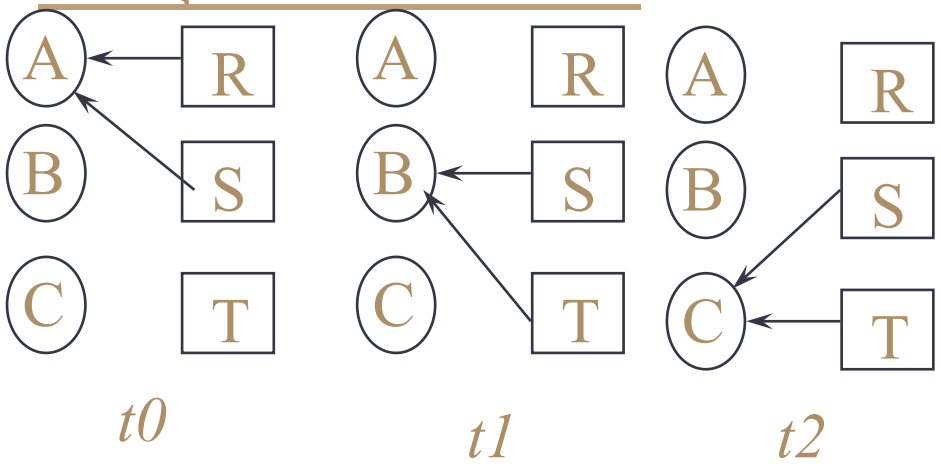


:deadlock envolvendo dois processos

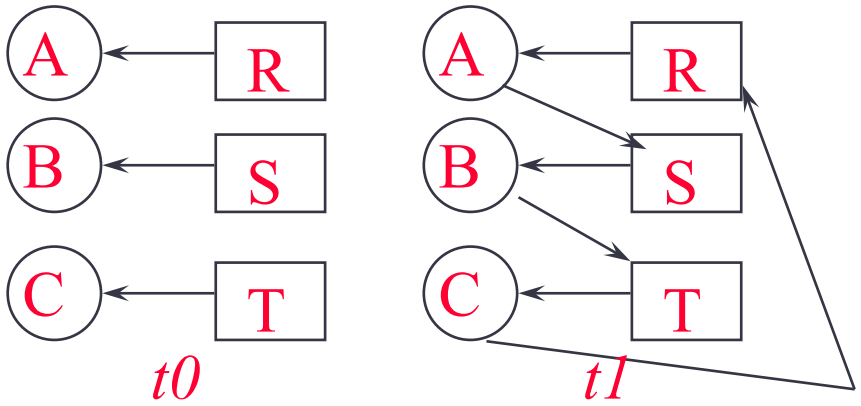
Deadlock Envolvendo 3 Processos

Processo A	Processo B	Processo C
Req(R)	Req(S)	Req(T)
Req(S)	Req(T)	Req(R)
Lib(R)	Lib(S)	Lib(T)
Lib(S)	Lib(T)	Lib(R)

Situação sem deadlock:



Situação com deadlock:



Estratégias para Tratar o Deadlock

- Ignorar o problema
- Detectar e recuperar o deadlock
- Evitar o deadlock
- Prevenir o deadlock

Ignorar o Problema

- Justificativa: Não vale a pena degradar a performance do sistema para tratar uma situação que ocorre com pouca frequência.
  - Ex: a maioria dos sistemas modernos
- Neste caso, quem trata o deadlock caso o mesmo ocorra?
  - O programador!!!

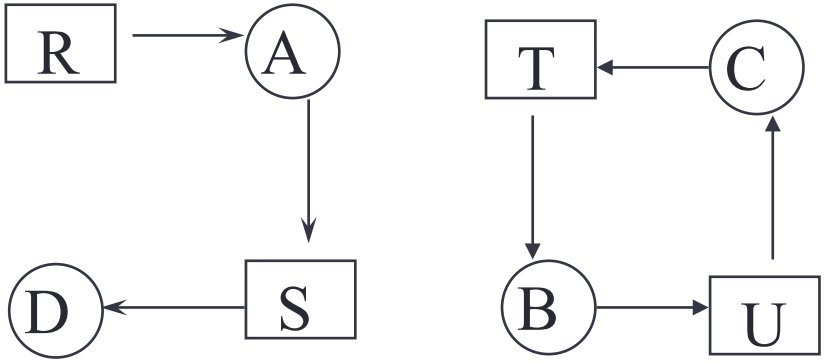
Detectar e Recuperar o Deadlock

- Justificativa: como existe a possibilidade de ocorrência do problema, o sistema operacional deve tratá-lo.
- Esta técnica é composta de duas etapas:
  - Detecção de Deadlock
  - Recuperação de Deadlock

Fase 1 - Detecção de Deadlocks

- Aproveitando o modelo de Holt, constrói-se o grafo de alocação de recursos.
- Se existir um ciclo, todos os processos que fazem parte deste ciclo estão em deadlock.
- Para detectar o ciclo, deve-se utilizar um algoritmo qualquer de detecção de ciclos em grafos dirigidos.

Fase 1 - Detecção de Deadlocks



Demonstração:

L = NULL;

L = [R, A, S, D]

L = NULL;

L = [C,T,B,U,C]

Fase 1 - Detecção de Deadlocks

- Quando detectar um deadlock?
  - A cada nova solicitação de recurso
  - Periodicamente
  - Quando a utilização da CPU for baixa

Fase 2 - Recuperação de Deadlocks

- Uma vez detectado, como eliminar o deadlock?
  - Preempção
  - Rollback
  - Eliminação de processos

Fase 2 - Recuperação de Deadlocks Preempção de Recursos

- Consiste em tomar o recurso do processo à força.
- A facilidade desta operação depende do recurso. Em geral, é uma operação complexa.
- A escolha do processo “preemptado”pode ser feita em função do recurso que ele possui.

Fase 2 - Recuperação de Deadlocks Rollback

- Gravação de um arquivo de log, atualizado periodicamente, que contem a imagem da memória ocupada pelo processo e o estado de alocação de recursos.
- Quando o deadlock ocorre, faz-se o rollback dos processos que contém recursos necessários à recuperação do deadlock até o instante imediatamente anterior à aquisição dos recursos. Os recursos em questão são entregues a um dos processos em deadlock.

Fase 2 - Recuperação de Deadlocks Eliminação de Processos

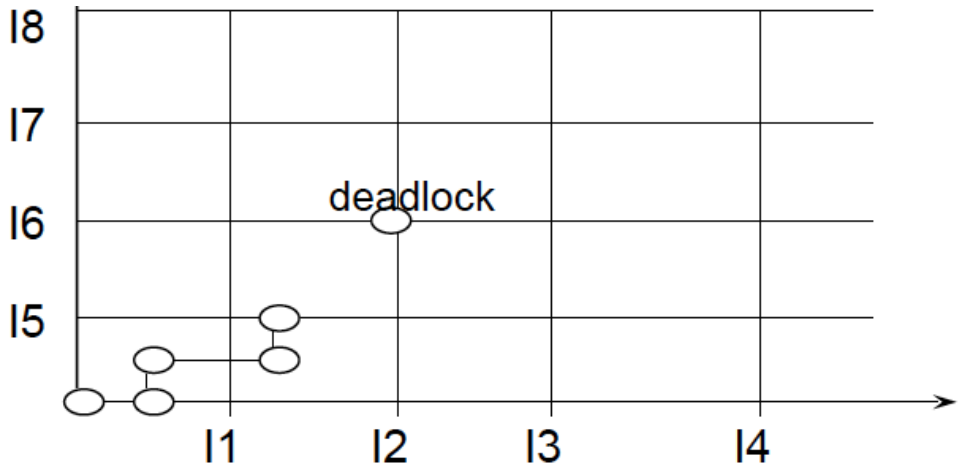
- Os processos que compõem o ciclo são eliminados um a um até que o ciclo seja quebrado.
- Os melhores processos a serem escolhidos são aqueles onde uma execução não depende das execuções anteriores.

Evitar o Deadlock

- A maioria dos algoritmos para evitar deadlocks baseia-se no conceito de estados seguros.
- Estado seguro: um estado é seguro se há uma maneira de satisfazer todas as requisições pendentes partindo dos processos em execução.
- Estado inseguro: não se pode garantir que as requisições serão satisfeitas.

Evitar o Deadlock (Exemplo)

Processo A		Processo B	
Req(impressora)	(I1)	Req(plotter)	(I5)
Req(plotter)	(I2)	Req(impressora)	(I6)
Lib(impressora)	(I3)	Lib(plotter)	(I7)
Lib(plotter)	(I4)	Lib(impressora)	(I8)



Evitar o Deadlock

- Algoritmo do Banqueiro (Dijkstra)
- Objetivo: garantir o credito aos correntistas do banco
- Entidades:
  - clientes: processos
  - unidade de crédito: recursos
  - banqueiro: sistema operacional
- Algoritmo:
  - Se uma solicitação de recursos leva a um estado seguro, atende-a.
  - Senão, a solicitação e adiada por um tempo (processo bloqueado)
  - solicitação segura: o banqueiro tem recursos suficientes para atender pelo menos um cliente.

Evitar o Deadlock (Algoritmo do Banqueiro)

	Possui	Max
A	1	6
B	1	5
C	3	4
D	4	7

Livre:1

estado:

	Possui	Max
A	2	6
B	1	5
C	2	4
D	4	7

Livre:1

estado:

- Foi generalizado para diversos tipos de recursos
- Problemas: não é um algoritmo utilizável, pois requer conhecimento do padrão futuro de requisição de recursos.

Prevenir o Deadlock

➔Como existem 4 condições necessárias para a ocorrência de deadlocks, uma maneira de impedir que estes ocorram e fazer com que uma destas 4 condições não seja nunca verdadeira.

- Impossibilitar a exclusão mútua
- Impossibilitar a posse e espera
- Impossibilitar a não-preempção
- Impossibilitar a espera circular

Prevenir o Deadlock
Impedir a Exclusão Mútua

- Restringir ao máximo o número de processos que podem solicitar um determinado recurso.
  - Impressoras: spool de impressão (o recurso so pode ser solicitado ao gerenciador de impressão)
  - Bloqueio de registros: SGBD
- Problemas: nem todos os recursos podem ser gerenciados por um processo único.

Prevenir o Deadlock
Impedir a Posse e Espera

- Técnica 1: fazer com que os processos solicitem todos os recursos que necessitam, antes da execução.
  - *Problema*: necessidade de conhecimento futuro, utilização não otimizada de recursos.
- Tecnica 2: um processo que solicita recursos deve liberar todos os recursos que adquiriu anteriormente para depois tentar adquirir de uma vez todos os recursos necessários.
  - *Problema*: programação mais complexa

Prevenir o Deadlock
Impedir a Não-Preempção

- Consiste em garantir que sempre haja uma maneira de retirar um recurso de um processo.
  - *Problema*: nem sempre é útil (ex: impressora) e pode conduzir a resultados inesperados (ex: bloqueio de registros).

Prevenir o Deadlock
Impedir a Espera Circular

- Impedir ciclos na alocação de recursos.
- Técnica: impor uma numeração global de todos os recursos do sistema e fazer com que os processos sigam esta ordem na requisição de recursos.
  - *Problema*: grande quantidade de recursos (posições em tabelas, registros bloqueados, etc).

Aspectos Relacionados ao Deadlock

- Preterição (postergação) indefinida (starvation): certas politicas de alocação de recursos, apesar de não causarem deadlock, podem fazer com que alguns processos nunca sejam atendidos (geralmente em situações de alta demanda de recursos). Este fenômeno e chamado postergação indefinida.
  - Exemplo: Politica de alocação de CPU: o processo escolhido para rodar e aquele que solicita mais I/O.
- Livelock:
  - Os processos não estão bloqueados porém estão em um ponto da execução do qual jamais sairão.

Deadlock - Conclusões

- O deadlock é um problema que pode ocorrer em um sistema onde:
  - os processos detém recursos exclusivamente e;
  - estando de posse de um recurso, o processo pode solicitar um outro recurso e;
  - o recurso só pode ser liberado pelo próprio processo.
  - Neste quadro, o deadlock é caracterizado pela presença de ciclos no grafo de alocação de recursos.

Deadlock - Conclusões

- Apesar de teoricamente possível, o tratamento de deadlocks e uma operação complexa.
- A maioria dos sistemas operacionais não utiliza nenhuma técnica de tratamento de deadlock.