Deadlocks

Professora: Lúcia Drummond

Deadlocks

- Recursos: hardware ou informação
- Preemptivo
- Não preemptivo

- Uso do Recurso:
  1. Pedido (Request ou Open)
  2. Uso
  3. Liberação

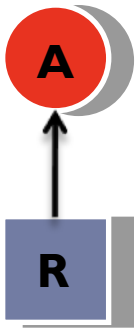
- "Um conjunto de processos está em deadlock se cada processo no conjunto está esperando por um evento que somente um outro processo do conjunto pode causar."

Deadlocks

- Condições
  1. Exclusão Mútua
  2. Obtenção e Espera
  3. Não preempção
  4. Espera Circular

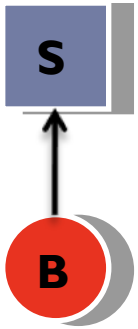
Deadlocks

- Modelo
  - Grafos: Processos são círculos e recursos são quadrados.
  - O recurso **R** está alocado ao processo **A**.



Deadlocks

- Modelo
  - O processo **B** está bloqueado esperando **S**.

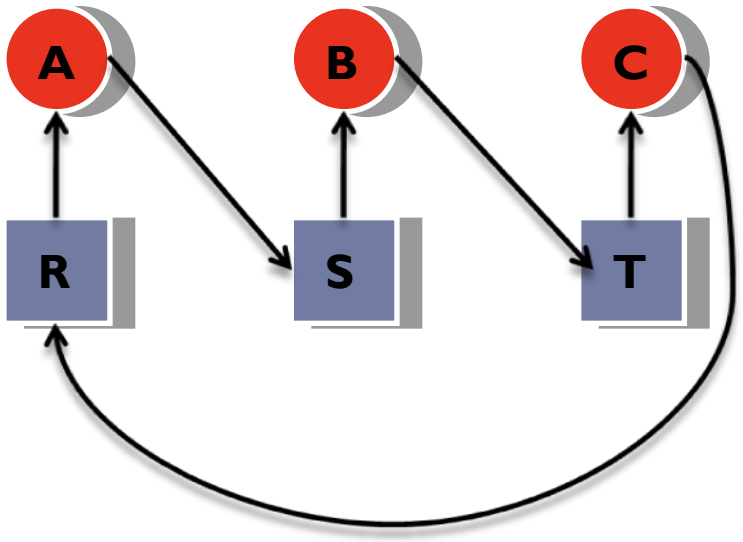


  - Exemplo:

Processo A	Processo B	Processo C
Pede R	Pede S	Pede T
Pede S	Pede T	Pede R
Libera R	Libera S	Libera T
Libera S	Libera T	Libera R

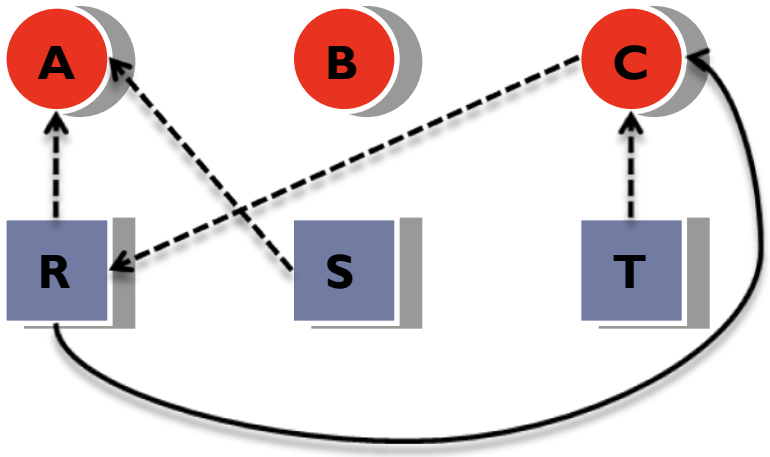
Deadlocks

- Modelo
  - Exemplo:
    1. A pede R
    2. B pede S
    3. C pede T
    4. A pede S
    5. B pede T
    6. C pede R
    7. deadlock



Deadlocks

- Modelo
  - Exemplo:
    1. A pede R
    2. C pede T
    3. A pede S
    4. C pede R
    5. A libera R
    6. A libera S



Deadlocks

- Estratégias
  1. Ignora o problema
  2. Detecção e Recuperação
  3. Evitar dinamicamente
  4. Prevenção

- Algoritmo do Avestruz
  - Unix: Número de entradas na tabela de processos é finita.
    - Exemplo: 100 entradas e 10 processos criam 12 processos. Depois que cada processo criou 9 outros, eles ficam em loop infinito.

Deadlocks

- Detecção e Recuperação de Deadlock
  - Com um tipo de recurso
    - Detecção de ciclos em grafos:

1-Para cada nó do grafo execute os seguintes passos:

2-Inicialize L como lista vazia e designe todos os arcos como não marcados.

3- Acrescente o nó atual ao final de L e verifique se o nó parece em L duas vezes. Se sim, o grafo contém um ciclo e o algoritmo termina.

4- Do dado nó, veja se existem quaisquer arcos de saída não marcados. Se sim, vá para o passo 5; senão vá para o passo 6.

Deadlocks

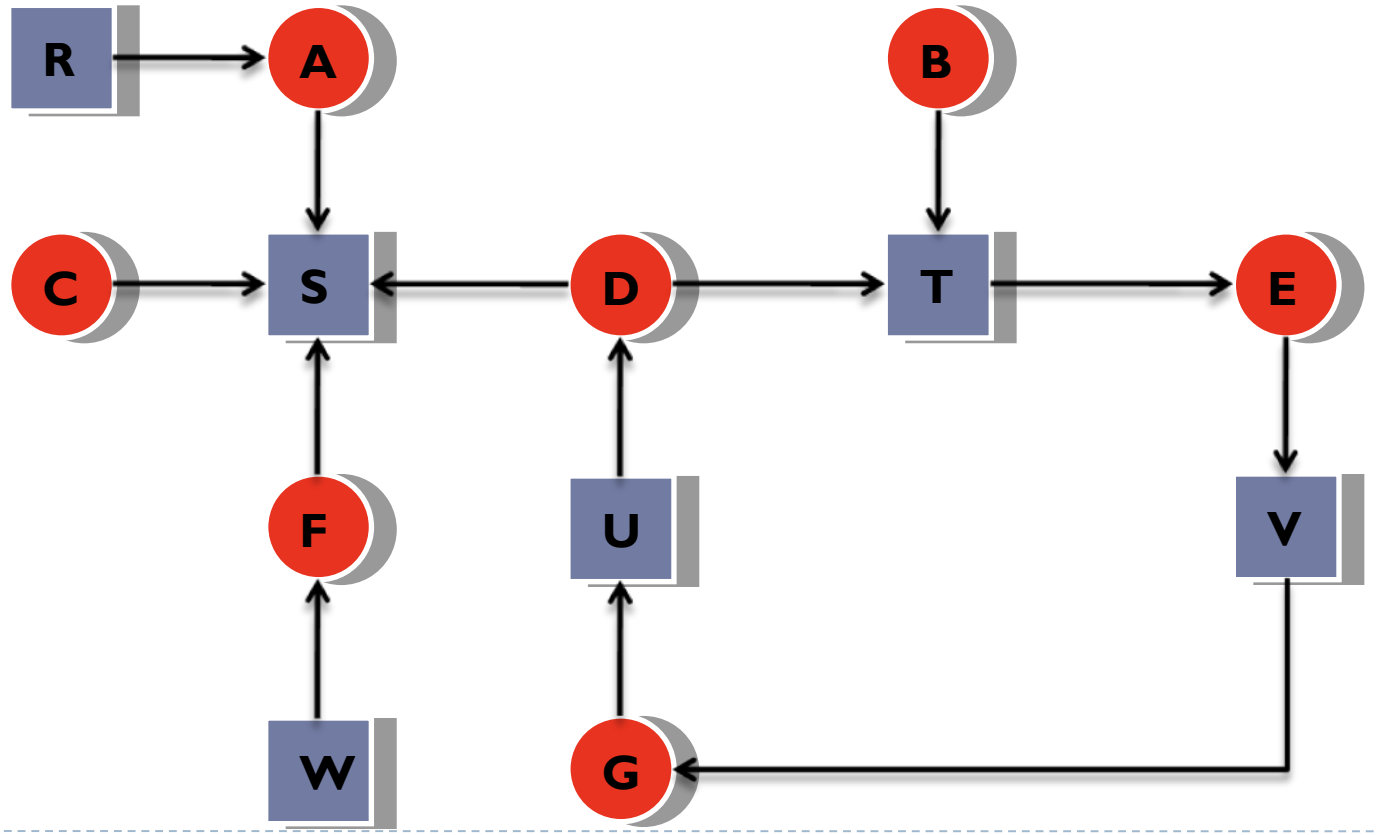
- Detecção e Recuperação de Deadlock
  - Com um tipo de recurso
    - Detecção de ciclos em grafos:

5- Pegue qualquer arco de saída não marcado aleatoriamente e marque-o. Então siga para o próximo nó corrente e vá para o passo 3

6- Dead end. Volte ao nó anterior, torne-o o atual e volte ao passo 4. Se este nó é o inicial, o grafo não contém ciclos e o algoritmo termina.

Deadlocks

- Detecção e Recuperação de Deadlock
  - Com um tipo de recurso



Deadlocks

- Detecção e Recuperação de Deadlock
  - Com múltiplos recursos
    - Classes de recursos = **m**
    - Vetor de recursos = **E** (**m** elementos)
    - Vetor de recursos disponíveis = **A**
    - Número de processos = **n**
    - Matriz de alocação atual = **C**
    - Matriz de pedidos = **R**

Deadlocks

- Detecção e Recuperação de Deadlock
  - Com múltiplos recursos
    - Exemplo:
      - $\mathbf{C_{i,j}}$ = número de instâncias do recurso **j** que são utilizadas por $\mathbf{P_i}$
      - $\mathbf{R_{i,j}}$ = número de instâncias do recurso **j** que $\mathbf{P_i}$ deseja.

$$\sum_{i=1}^{m} (C_{i,j} + A_j) = E_j$$

Deadlocks

- Detecção e Recuperação de Deadlock
  - Recuperação de deadlock
    - Preempção: tomar o recurso de um processo temporariamente e dá-lo a outro processo.
      - Exemplo: Mainframe: Intervenção manual.
    - Rollback e checkpointing (memory image estados dos processos): quando o deadlock é detectado, um processo pode retornar a um determinado ponto.
    - Matar processos.

Deadlocks

- Detecção e Recuperação de Deadlock
  - Algoritmo de detecção
    1. Procure um processo não marcado **P_i** para o qual a i-ésima linha de **R** é menor que **A**.
    1. Se tal processo é achado, acrescente a i-ésima linha de **C** para **A**, marque o processo **C** e volte para o passo 1.
    1. Se tal processo não existe, o algoritmo termina.

    - Quando o algoritmo termina, todos os processos não marcados estão em deadlock.

Deadlocks

- Detecção e Recuperação de Deadlock
  - Algoritmo de detecção
    - Exemplo:

$$E = (4 \quad 3 \quad 2 \quad 1) \qquad A = (2 \quad 1 \quad 0 \quad 0)$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Deadlocks

- Evitar Deadlocks
  - O sistema é capaz de decidir se obter um recurso é seguro ou não e somente faz a alocação quando for seguro.

  - Estados seguros e não seguros
    - Seguro: O sistema não está em deadlock e existe uma forma de satisfazer todos os pedidos pendentes.

Deadlocks

- Evitar Deadlocks
  - Algoritmo do banqueiro
    - O algoritmo do banqueiro considera cada pedido quando ocorre e verifica se atendê-lo leva a um estado seguro. Se sim, o pedido é atendido, senão o pedido é adiado.
    - Para verificar se um estado é seguro:
      - O banqueiro examina se ele tem recursos suficientes para satisfazer algum cliente.
      - Se sim, ele assume que estes recursos serão devolvidos, e testa o limite de algum outro cliente.
      - Se todos os empréstimos puderem ser pagos, o estado é seguro e o pedido inicial pode ser atendido.

Deadlocks

- Evitar Deadlocks
  - Algoritmo do banqueiro para múltiplos recursos

1. Procure uma linha em **R**, cujas necessidades de recurso sejam menores ou iguais a **A**. Se tal linha não existir, o sistema eventualmente estará em deadlock já que nenhum processo poderá executar até terminar.

1. Assuma que o processo da linha escolhida peça todos os recursos que precisa e termine. Marque este processo como terminado e some todos os seus recursos ao vetor **A**.

1. Repita os passos 1 e 2 até que ou todos os processos estejam marcados como terminados, neste caso o estado inicial é seguro ou até que um deadlock ocorra.

Deadlocks

- Prevenção de Deadlock
  1. Exclusão Mútua
     - Printer – Spool
     - Espaço de disco de spool – deadlock
  2. Obtenção e espera
     - Os processos obtêm os recursos antes de começarem a execução.
  3. Não preempção
     - Problema!!!
  4. Espera Circular
     - Numerar os recursos.
     - Processos pedirem em ordem.

Deadlocks

- Prevenção de Deadlock
  - Outras considerações

1. Locking de duas fases
   - Obter todos os recursos antes de terminar.
2. Non-resources
   - Semáforos
3. Starvation