DEADLOCKS

- **Um Deadlock (ou "bloqueio perpétuo" ou "impasse") pode ser provocado quando cada processo de um conjunto está esperando por um evento que apenas outro processo do conjunto pode causar**

- **Deadlocks podem ocorrer quando processos podem conseguir acesso exclusivo a dispositivos, arquivos e similares**

- **Estes objetos podem ser chamados genericamente de recursos**

Recursos

- Exemplos de Recursos computacionais
  - impressoras
  - Unidades de fitas
  - Tabelas do sistema de arquivos
- Processos precisam acessar os recursos em ordem razoável
- Suponha que um processo mantenha um recurso A e requisita um recurso B
  - Ao mesmo tempo um outro processo mantém B e requisita A
  - Ambos são bloqueados e permanecerão assim

Recursos (1)

- Deadlocks podem ocorrer quando ...
  - É garantido acesso exclusivo aos recursos
  - Nós nos referimos a estes dispositivos como recursos
- Recursos Preemptíveis
  - Podem ser retomados dos processos sem nenhum efeito
- Recursos não preemptíveis
  - Causarão falha no processo se forem retomados

Recursos (2)

- Sequência de eventos requirida para usar um recurso
  1. Requisita o recurso
  2. Usa o recurso
  3. Libera o recurso
- Deve esperar se a requisição for negada
  - O processo requerente pode ser bloqueado
  - Pode falhar, retornando um código de erro

Gerência de E/S

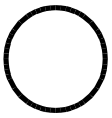
Diagramas de Processos X Recursos

- **O estudo dos bloqueios perpétuos pode ser bastante facilitado pelo uso de diagramas de alocação de recursos**

processo

Processo requisitando recurso

recurso



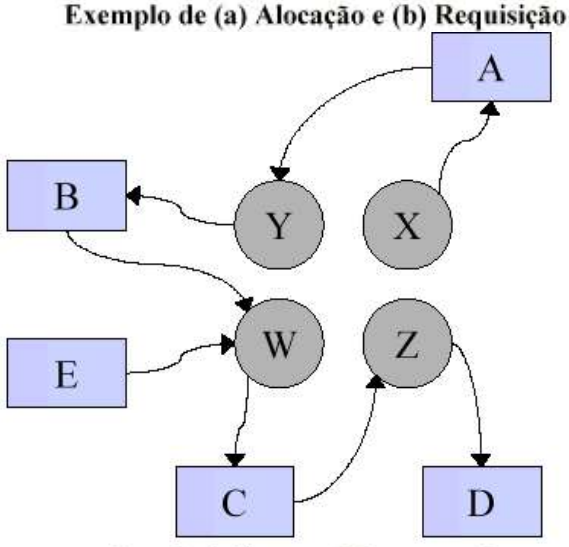
Recurso alocado pelo processo

Exemplo de alocação e solicitação de recursos

Tabela de Alocação de Recursos	
Recurso	**Processo**
X	A
Y	B
W	C
Z	D

Tabela de Requisição de Recursos	
Processo	**Recurso**
A	Y
B	W
C	Z
E	W

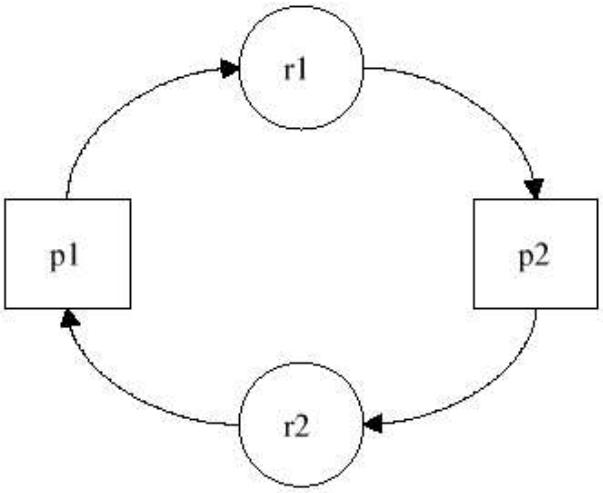
Exemplo de (a) Alocação e (b) Requisição de Recursos



Exemplo de Diagrama de Processos x Recursos

Gerência de E/S

Utilizando o diagrama de alocação de recursos, um deadlock envolvendo dois processos e dois recursos, seria representado:



Fica clara a situação de formação de uma cadeia circular ou caminho fechado no pedido e alocações de recursos no sistema

Gerência de E/S

Condições para Ocorrência de Deadlocks

- **Existem 4 condições para ocorrência de um deadlock:**
  - **Processos exigem controle exclusivo sobre o recurso que solicitam (exclusão mútua)**
  - **Processos mantêm alocados recursos enquanto solicitam novos recursos (condição de espera por recurso)**
  - **Recursos não podem ser retirados dos processos, enquanto estes não finalizarem o uso (ausência de preemptividade)**
  - **Forma-se uma cadeia circular de processos, cada um solicitando o recurso alocado ao próximo da cadeia (espera circular)**

Estratégias para o tratamento de Deadlocks

- Existem 4 alternativas, segundo Tanenbaum, para resolver problemas de deadlock:
  - Ignorar o problema (algoritmo do avestruz)
  - Detecção e recuperação de deadlocks
  - Prevenção dinâmica através de procedimentos cuidadosos de alocação
  - Prevenção estrutural através da negação de uma ou mais das quatro condições de ocorrência

Algoritmo do Avestruz

- Como os deadlocks podem ser raros, uma opção é ignorar o problema
- Exs: Windows, Linux e boa parte dos SOs atuais fazem isto.

Detecção e Recuperação de Deadlocks

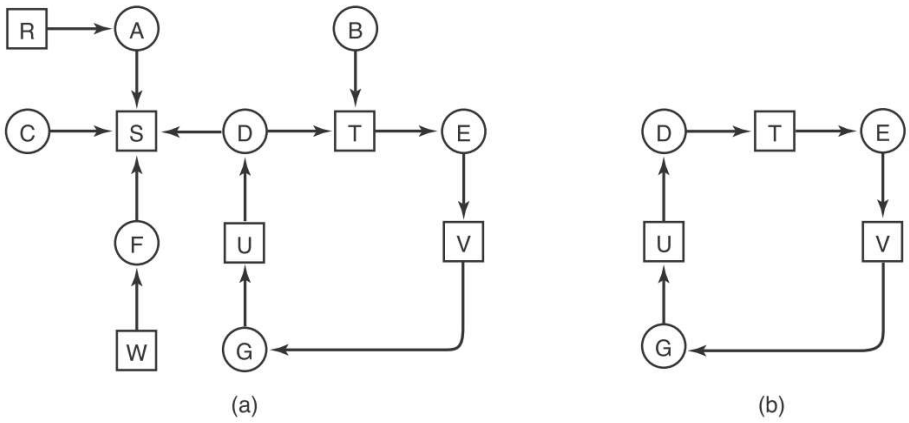
Quando essa técnica é usada, o sistema não tenta evitar a ocorrência dos impasses.

Em vez disso, ele os deixa ocorrer, tenta detectá-los quando acontecem e então toma alguma medida para recuperar-se após o fato.

Detecção e Recuperação de deadlocks

Detecção: Uma abordagem - construir um grafo de recursos

(a) Um grafo de recursos. (b) Um ciclo extraído de (a).



Detecção e Recuperação de deadlocks

Recuperação após deadlock

Recuperação mediante preempção

A capacidade de tirar um recurso de um processo é algo altamente dependente da natureza do recurso.
A recuperação dessa maneira é com frequência difícil ou impossível

Recuperação mediante retrocesso

Processos geram pontos de salvaguarda

Recuperação: processo que tem um recurso necessário é Retrocedido até o ponto anterior a obtenção daquele recurso

Detecção e Recuperação de deadlocks

Recuperação mediante eliminação de processos

A maneira mais bruta de eliminar um impasse, mas também a mais simples, é matar um ou mais processos.

Sempre que possível, é melhor matar um processo que pode ser reexecutado desde o início sem efeitos danosos.

Prevenção dinâmica através de procedimentos cuidadosos de alocação

Na maioria dos sistemas: os recursos são solicitados um de cada vez.

O sistema precisa ser capaz de decidir se conceder um recurso é seguro ou não e fazer a alocação somente quando for seguro

Os principais algoritmos para evitar impasses são baseados no conceito de **estados seguros**.

Diz-se de um **estado que ele é seguro** se existir alguma ordem de escalonamento na qual todos os processos puderem ser executados até sua conclusão mesmo que todos eles subitamente solicitem seu número máximo de recursos imediatamente.

Dificuldade: na prática é difícil se saber, de antemão, quais suas necessidades máximas de recursos

Estratégias para o tratamento de Deadlocks

Prevenção estrutural através da negação de uma ou mais das quatro condições de ocorrência (tabela abaixo)

Condição	Aproximação
Exclusão Mútua	Colocar todos os recursos do sistema em *spool*
Retenção e Espera	Exigir a alocação inicial de todos os recursos necessários
Sem Preemptividade	Retirada de recursos dos processos
Espera Circular	Ordenação numérica dos recursos

Espera Circular
- saída simples – ter uma regra dizendo que um processo tem o direito de um único recurso de cada vez
- Uma forma de assegurar que essa condição jamais ocorra é impor uma ordem absoluta a todos os tipos de recursos e requerer que cada processo solicite recursos em uma ordem de enumeração crescente.

F (drive de fita) = 1
F (drive de disco) = 5
F (impressora) = 12