Sistemas Operacionais

DeadLock

Edeyson Andrade Gomes

www.edeyson.com.br

Roteiro da Aula

- DeadLock
  - Metas
  - Algoritmos

DeadLock

- Um estado de deadlock ocorre quando dois ou mais processos estão esperando por um evento que só pode ser gerado por algum dos processos em espera;
  - Processo espera por um evento que nunca ocorrerá;
  - Cada processo do conjunto em *deadlock* está esperando por um recurso que foi entregue a outro processo do mesmo conjunto.
  - Espera circular por recursos.

DeadLock

- Recurso pode ser removível e não removível.
- Geralmente deadlock surge quando há recursos não removíveis envolvidos.

DeadLock

- Condições necessárias para ocorrência do DeadLock:
  1. Exclusão Mútua;
  2. Posse e Espera;
  3. Não Preempção;
  4. Espera Circular.
- As 4 condições necessitam ocorrer em conjunto
  - Se ao menos uma não ocorrer, não há DeadLock

DeadLock

- Condições necessárias para ocorrência do DeadLock:
  1. Exclusão Mútua;
     1. Apenas um processo por vez pode alocar e manipular um recurso
        1. Recurso de uso exclusivo
           1. Impressora, CD/DVD para gravação
  2. Posse e Espera;
  3. Não-Preempção;
  4. Espera Circular.

DeadLock

- Condições necessárias para ocorrência do DeadLock:
  1. Exclusão Mútua;
  2. Posse e Espera;
     1. Um processo, de posse de um recurso, pode solicitar novos recursos
  3. Não-Preempção;
  4. Espera Circular.

DeadLock

- Condições necessárias para ocorrência do DeadLock:
  1. Exclusão Mútua;
  2. Posse e Espera;
  3. Não-Preempção;
     1. Um recurso não pode ser removido explicitamente do processo
        1. Ex.: Impressora, CD (Gravação)
  4. Espera Circular.

DeadLock

- Condições necessárias para ocorrência do DeadLock:
  1. Exclusão Mútua;
  2. Posse e Espera;
  3. Não-Preempção;
  4. Espera Circular.
     1. Ocorre CICLO no GRAFO de alocação

DeadLock

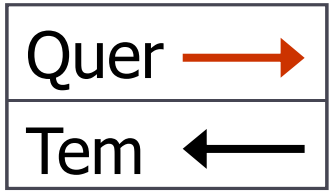
- Como usar recursos:
  1. Solicitar Recurso ao S.O.
  2. Se o Recurso estiver LIVRE
     ALOCAR RECURSO
  3. Senão:
     BLOQUEAR PROCESSO
  4. Após Uso:
     1. Processo LIBERA Recurso

DeadLock

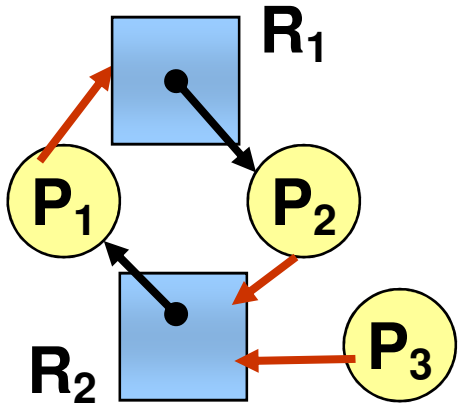
- Como usar recursos:
  1. Solicitar Recurso ao S.O. DOWN(mutex)
  2. Se o Recurso estiver LIVRE se mutex = 1
     ALOCAR RECURSO
  3. Senão: mutex = 0
     BLOQUEAR PROCESSO
  4. Após Uso:
     1. Processo LIBERA Recurso **UP(mutex)**

Grafo de Alocação de Recursos

Deadlock
- Imprevisível
- Depende da ordem dos Eventos



Eventos:
P1 requisitou R2
P2 requisitou R1
P1 requisitou R1
P2 requisitou R2
P3 requisitou R2

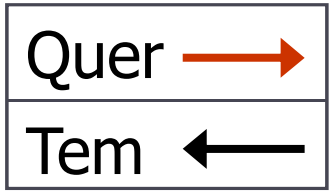


Grafo de Alocação de Recursos

Alocação de Recursos

Implementação com Semáforos?

Com Monitores?



Eventos:

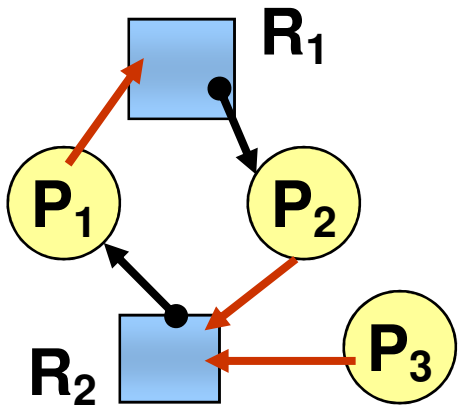
P1 requisitou R2

P2 requisitou R1

P1 requisitou R1

P2 requisitou R2

P3 requisitou R2

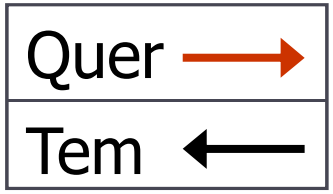


Grafo de Alocação de Recursos

Deadlock

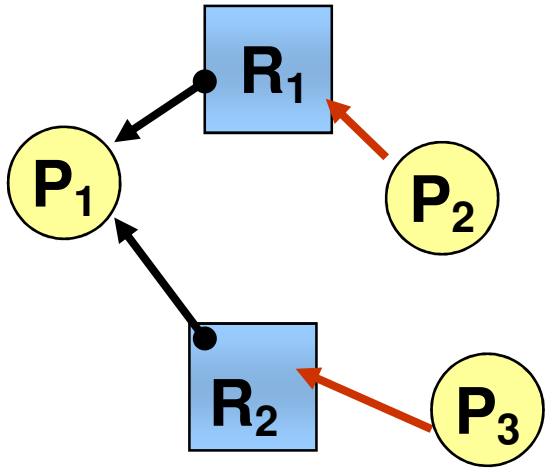
Imprevisível

Depende da ordem dos Eventos



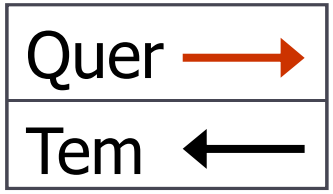
Eventos:
P1 requisitou R2
P1 requisitou R1

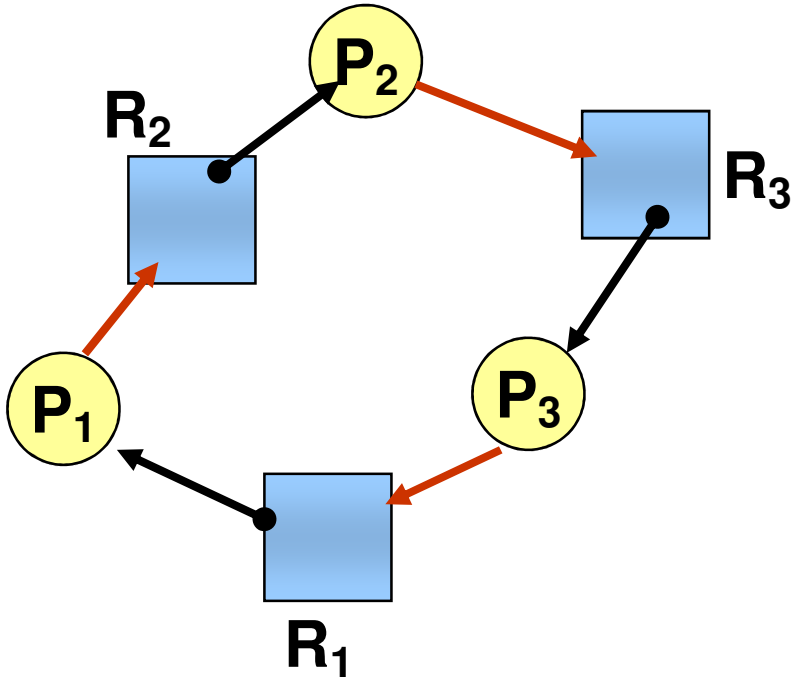
P2 requisitou R1
P3 requisitou R2



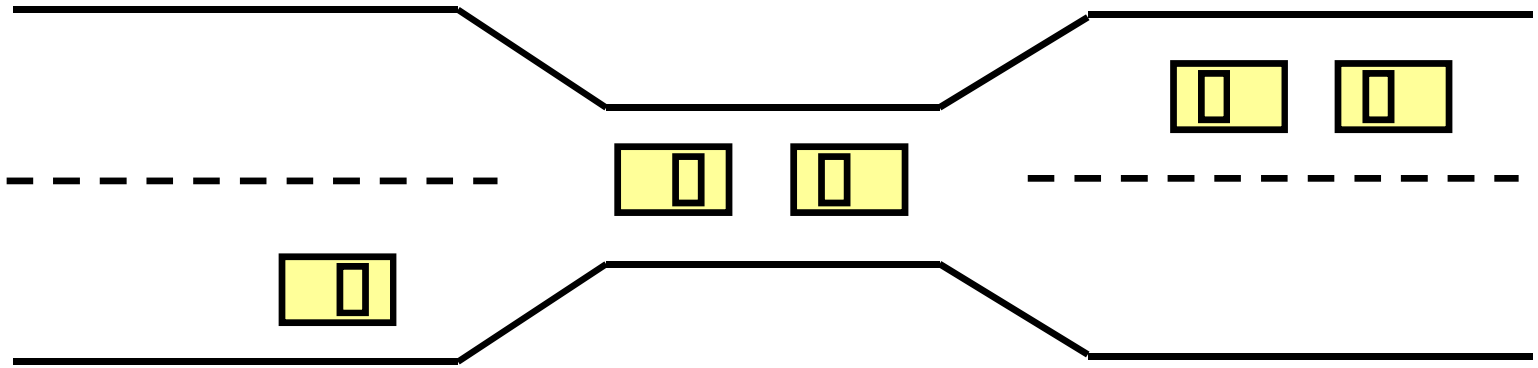
Grafo de Alocação de Recursos

Grafo com DeadLock
Ciclo Existente





Exemplo Comparativo



- Exemplo da ponte (recurso):
  - Só permite tráfego em um sentido por vez.

Soluções

- Ignorar o problema ("Algoritmo do Avestruz");
  - Linux, Unix, Windows
- Detectar e Recuperar;
- Evitar dinamicamente sua ocorrência, com alocação cuidadosa de recursos;
- Impedir o *Deadlock* através da negação de pelo menos uma das 4 condições necessárias para sua ocorrência.

Detectar e Recuperar

- Dois Modelos:
  - Um recurso de cada tipo
  - Vários recursos de cada tipo

Deadlock

Detecção com ***um recurso de cada tipo***

Detectar e Recuperar

- Detecção com um recurso de cada tipo
  - Monta Grafo
    - Se houver ciclo
      - DETECTA DEADLOCK
  - Analisar grau de multiprogramação
    - Taxa de bloqueio e tempo de espera

- Problema:
  - Quando montar o grafo?

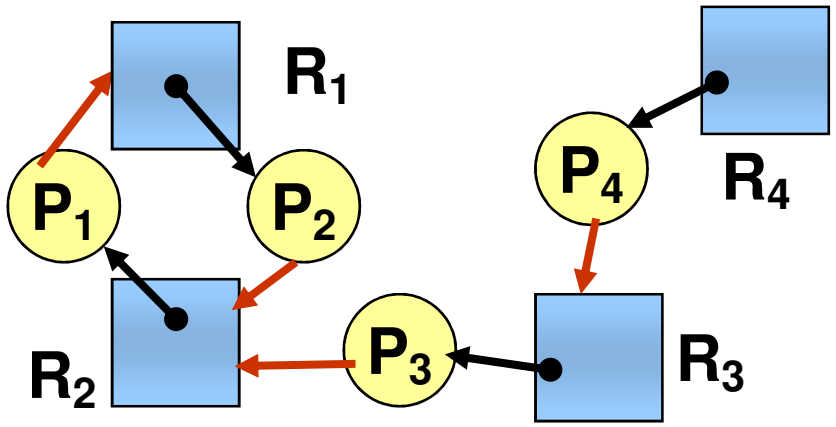
Detectar e Recuperar

- Detecção com um recurso de cada tipo
  - Problema:
    - Quando montar o grafo?
      - A cada requisição de recurso (I)
      - Em intervalos fixos (II)
    - Qual o custo de cada um?
      - Overhead
        - A solução (I) tem maior overhead.
        - Detecta imediatamente o Deadlock
        - E a solução (II)?

Detectar e Recuperar

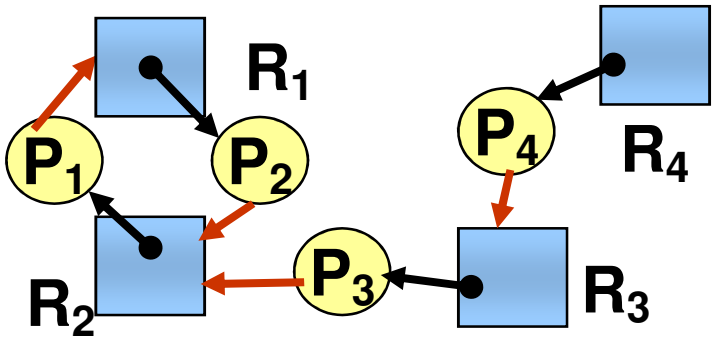
- Recuperação com um recurso de cada tipo
  - Problema:
    - O Deadlock ocorreu. Como recuperar?

Eventos:
P1 req R2
P2 req R1
P3 req R3
P4 req R4
P1 req R1
P2 req R2
P3 req R2
P4 Req R3



Detectar e Recuperar

- Recuperação com um recurso de cada tipo
  - Soluções:
    - Extermínio
      - ***Mata um processo no Ciclo***
    - Volta ao Passado
      - Rollback em processo do Ciclo até liberar recurso



Detectar e Recuperar

- Recuperação com Extermínio
  - ***Critérios de escolha da Vítima no Ciclo***:
    - Aleatório
    - Prioridade
    - Tempo de CPU
    - Número de Recursos Alocados

- Detecção x Recuperação

Detectar e Recuperar

- Recuperação com Volta ao Passado
  - P1 ---|R2|------|------|R1|------|-------|----
  - P2 ---|------|R1|------|------|R2|-------|----
  - P2 é alvo
    - Desfazer P2 até antes de alocar R1

Detectar e Recuperar

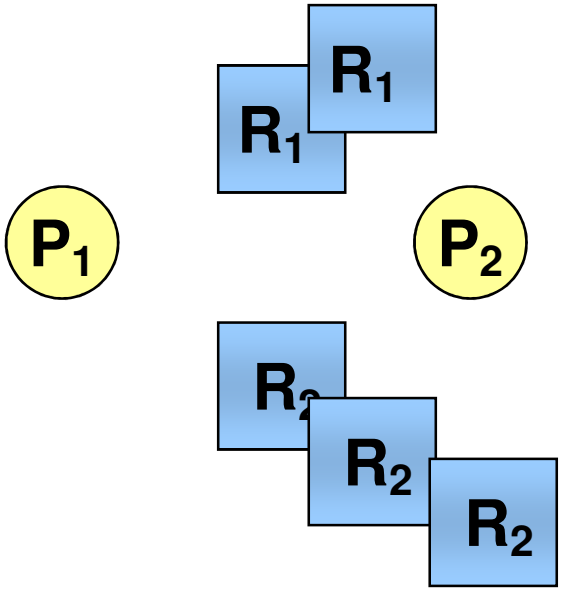
- Recuperação com Volta ao Passado
  - P2 é alvo
    - Desfazer P2 até antes de alocar R1
    - P1 ---|R2|------|------|R1|------|-------|----
    - P2 ---|------|
  - Custo.
    - Quem implementa?
      - SGBD

Deadlock

Detecção com ***Vários Recursos de Cada Tipo***

Detectar e Recuperar

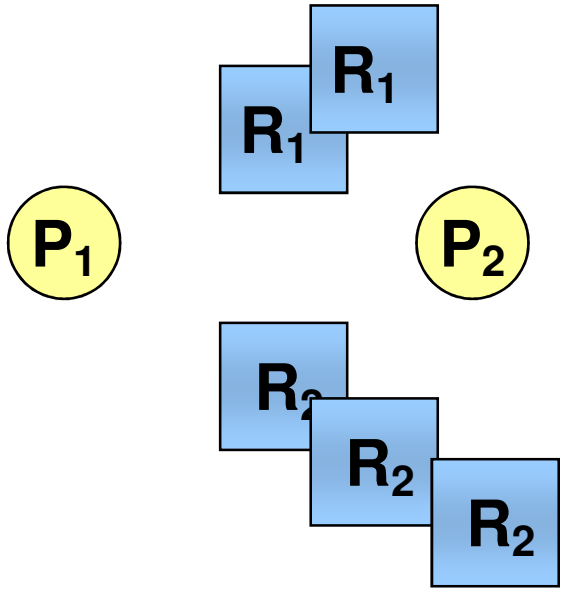
- Detecção e Recuperação com Vários Recursos de Cada Tipo
  - Suponha que eu tenho 2 R1 e 3 R2
  - Como montar o Grafo?



Detectar e Recuperar

- Alocação
  - Suponha que eu tenho 2 R1 e 3 R2

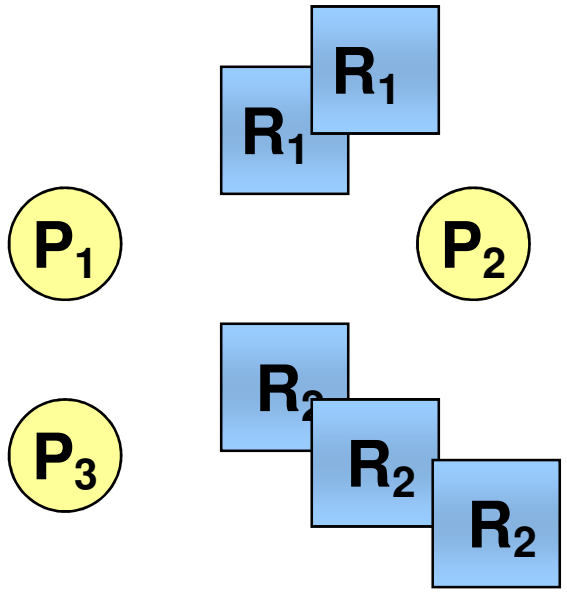
Evento	Recursos Disponíveis	
P1 req R1	[R1 = 1 e R2 = 3]	
P2 req R1	[R1 = 0 e R2 = 3]	
P1 req R2	[R1 = 0 e R2 = 2]	
P1 req R2	[R1 = 0 e R2 = 1]	
P2 req R2	[R1 = 0 e R2 = 0]	
P1 req R1	[R1 = 0 e R2 = 0]	P1 → BL
P2 req R2	[R1 = 0 e R2 = 0]	P2 → BL



Detectar e Recuperar

- Alocação
  - Suponha que eu tenho 2 R1 e 3 R2

Evento	Recursos Disponíveis	
P1 req R1	[R1 = 1 e R2 = 3]	
P2 req R1	[R1 = 0 e R2 = 3]	
P1 req R2	[R1 = 0 e R2 = 2]	
P3 req R2	[R1 = 0 e R2 = 1]	
P2 req R2	[R1 = 0 e R2 = 0]	
P1 req R1	[R1 = 0 e R2 = 0]	P1 ➔ BL
P2 req R2	[R1 = 0 e R2 = 0]	P2 ➔ BL



Não há DeadLock

Detectar e Recuperar

- Solução de Detecção
  - ***Uso de Vetores e Matrizes***
  - Vetor de Recursos Existentes (VRE)
    - A posição i do vetor representa a quantidade de recursos i
    - [p q r]
      - p recursos to tipo 1
      - q recursos do tipo 2
      - r recursos do tipo 3
  - VRE = [3 2 5]
    - R1 = Fitas
    - R2 = CD
    - R3 = Impressoras

Detectar e Recuperar

- Solução de Detecção
  - Uso de Vetores e Matrizes
  - ***Vetor de Recursos Disponíveis (VRD)***
    - A posição i do vetor representa quantos recursos i não estão Alocados
    - [m n o]
      - m recursos to tipo 1 disponíveis
      - n recursos do tipo 2 disponíveis
      - o recursos do tipo 3 disponíveis
  - VRD = [1 2 3]
    - R1 = Fita
    - R2 = CD
    - R3 = Impressoras

Detectar e Recuperar

- Solução de Detecção
  - Uso de Vetores e Matrizes
  - ***Matriz de Recursos Alocados (MRA)***
    - Linha representa um processo
    - Coluna representa um recurso

R1	R2	R3
1	0	2
0	1	1
0	0	1

MRA

P1 tem 1 R1, 0 R2 e 2 R3

P2 tem 0 R1, 1 R2 e 1 R3

P3 tem 0 R1, 0 R2 e 1 R3

Detectar e Recuperar

- Solução de Detecção
  - Uso de Vetores e Matrizes
  - ***Matriz de Recursos Requisitados (MRR)***
    - Linha representa um processo
    - Coluna representa um recurso

	R1	R2	R3		
	1	0	2		P1 requer 1 R1, 0 R2 e 2 R3
	0	1	1	MRR	P2 requer 0 R1, 1 R2 e 1 R3
	0	0	1		P3 requer 0 R1, 0 R2 e 1 R3

Detecção

VRE [2 3 1] VRD [2 3 1]

MRA $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

MRR $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

Eventos:

Detecção

VRE [2 3 1] VRD [2 3 1]

MRA $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

MRR $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

Eventos:

P1 requisita 1 R1 e 1 R2

SO:

Existem recursos livres?

Analisa VRD.

Detecção

VRE [2 3 1] VRD [1 2 1]

MRA $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

MRR $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

Eventos:

P1 requisita 1 R1 e 1 R2

SO:

Existem recursos livres?

Analisa VRD.

Aloca Recursos

Detecção

VRE [2 3 1] VRD [1 2 1]

MRA $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

MRR $\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$

Eventos:

P2 requisita 1 R1 e 1 R3

SO:

Existem recursos livres?

Analisa VRD.

Detecção

VRE [2 3 1] VRD [0 2 0]

MRA $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$

MRR $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

Eventos:

P2 requisita 1 R1 e 1 R3

SO:

Existem recursos livres?

Analisa VRD.

Aloca Recursos

Detecção

VRE [2 3 1] VRD [0 2 0]

MRA $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$

MRR $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 2 & 1 \end{pmatrix}$

Eventos:

P3 requisita 2 R2 e 1 R3

SO:

Existem recursos livres?

Analisa VRD.

Detecção

VRE [2 3 1] VRD [0 0 0]

MRA

1	1	0
1	0	1
0	2	0

MRR

0	0	0
0	0	0
0	0	1

Eventos:

P3 requisita 2 R2 e 1 R3

SO:

Existem recursos livres?

Analisa VRD.

Aloca Recursos

P3 ➔ Bl aguardando 1 R3

Não existe Deadlock

Detecção

VRE [2 3 1] VRD [0 0 0]

MRA $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \end{pmatrix}$

MRR $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Eventos:

P1 requisita 1 R1

SO:

Existem recursos livres?

Analisa VRD.

P1 ➔ Bl aguardando 1 R1

Não existe Deadlock

Detecção

VRE [2 3 1] VRD [0 0 0]

MRA $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \end{pmatrix}$

MRR $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Eventos:

P2 requisita 1 R2

SO:

Existem recursos livres?

Analisa VRD.

P2 ➔ Bl aguardando 1 R2

Existe Deadlock

Detectar e Recuperar

- Alocação
  - Suponha que eu tenho 2 R1 e 3 R2

Evento	Recursos Disponíveis
P1 req R1	[R1 = 1 e R2 = 3]
P2 req R1	[R1 = 0 e R2 = 3]
P1 req R2	[R1 = 0 e R2 = 2]
P1 req R2	[R1 = 0 e R2 = 1]
P2 req R2	[R1 = 0 e R2 = 0]
P1 req R1	[R1 = 0 e R2 = 0] P1 ➔ BL
P2 req R2	[R1 = 0 e R2 = 0] P2 ➔ BL

Detectar e Recuperar

- Alocação
  - Suponha que eu tenho 2 R1 e 3 R2

VRE = [2 3] VRD = [2 3]

VRE = [2 3] VRD = [0 0]

$$MRA \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \quad MRR \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Evento	Recursos Disponíveis
P1 req R1	[R1 = 1 e R2 = 3]
P2 req R1	[R1 = 0 e R2 = 3]
P1 req R2	[R1 = 0 e R2 = 2]
P1 req R2	[R1 = 0 e R2 = 1]
P2 req R2	[R1 = 0 e R2 = 0]
P1 req R1	[R1 = 0 e R2 = 0] P1 → BL
P2 req R2	[R1 = 0 e R2 = 0] P2 → BL

Detectar e Recuperar

- Qual o custo desse processo de Detecção?
  - Montar Matrizes e Vetores
  - Análise
    - Overhead

Deadlock

Como Evitar

Como Evitar Deadlock

- Algoritmo do Banqueiro
  - Empréstimo X Reserva
  - Qual o $ que o banco empresta?

  - Estado Seguro
    - Probabilidade de Deadlock = 0%

Evitar DeadLock

- ***Uso de Vetores e Matrizes***
  - Vetor de Recursos Existentes (VRE)
    - A posição i do vetor representa a quantidade de recursos i
    - [n1 n2 n3]
      - n1 recursos to tipo 1
      - n2 recursos do tipo 2
      - n3 recursos do tipo 3
  - VRE = [3 2 5]
    - R1 = 3 Fitas
    - R2 = 2 CD
    - R3 = 5 Impressoras

Evitar DeadLock

- Uso de Vetores e Matrizes
  - ***Vetor de Recursos Disponíveis (VRD)***
    - A posição i do vetor representa quantos recursos i não estão Alocados
    - [n1 n2 n3]
      - n1 recursos do tipo 1 disponíveis
      - n2 recursos do tipo 2 disponíveis
      - n3 recursos do tipo 3 disponíveis
  - VRD = [1 2 3]
    - R1 = 1 Fita
    - R2 = 2 CD
    - R3 = 3 Impressoras

Evitar DeadLock

- Evitar
  - Uso de Vetores e Matrizes
  - ***Matriz de Recursos Alocados (MRA)***
    - Linha representa um processo
    - Coluna representa um recurso

R1	R2	R3
1	0	2
0	1	1
0	0	1

MRA

P1 tem 1 R1, 0 R2 e 2 R3

P2 tem 0 R1, 1 R2 e 1 R3

P3 tem 0 R1, 0 R2 e 1 R3

Evitar DeadLock

- Evitar
  - Uso de Vetores e Matrizes
  - ***Matriz de Recursos Requisitados (MRR)***
    - Linha representa um processo
    - Coluna representa um recurso

R1	R2	R3		
1	0	2		P1 requer 1 R1, 0 R2 e 2 R3
0	1	1	MRR	P2 requer 0 R1, 1 R2 e 1 R3
0	0	1		P3 requer 0 R1, 0 R2 e 1 R3

Evitar DeadLock

- Evitar
  - Uso de Vetores e Matrizes
  - ***Matriz de Recursos Possivelmente Necessários (MRPN)***
    - Informados no início da execução do processo

R1	R2	R3
2	1	2
1	2	3
2	2	4

MRPN

P1 pode requerer 2 R1, 1 R2 e 2 R3
P2 pode requerer 1 R1, 2 R2 e 3 R3
P3 pode requerer 2 R1, 2 R2 e 4 R3

Evitar DeadLock

- Restrições
  - A alocação de recursos depende de estado seguro
  - O processo necessita informar ao sistema quantos recursos de cada tipo ele pode requerer

  - Alocação Criteriosa de Recursos

Evitar DeadLock

VRE [5 3 2]

VRD [5 3 2]

Eventos:

MRA $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

MRR $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

MRPN $\begin{pmatrix} 2 & 2 & 1 \\ 3 & 2 & 2 \\ 1 & 2 & 2 \end{pmatrix}$

Evitar DeadLock

VRE [5 3 2]

VRD [5 3 2]

Eventos: P1 solicita 1R1, 1R2, 1R3

MRA

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

MRR

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

MRPN

$$\begin{pmatrix} 2 & 2 & 1 \\ 3 & 2 & 2 \\ 1 & 2 & 2 \end{pmatrix}$$

Evitar DeadLock

VRE [5 3 2] VRD [4 2 1]

Eventos: P1 solicita 1R1, 1R2, 1R3

MRA
$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

MRR
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

MRPN
$$\begin{pmatrix} 2 & 2 & 1 \\ 3 & 2 & 2 \\ 1 & 2 & 2 \end{pmatrix}$$

Evitar DeadLock

VRE [5 3 2] VRD [4 2 1]

MRA
$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

MRR
$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

MRPN
$$\begin{pmatrix} 2 & 2 & 1 \\ 3 & 2 & 2 \\ 1 & 2 & 2 \end{pmatrix}$$

Eventos:

P1 solicita 1R1, 1R2, 1R3

P2 solicita 1R1, 1R3

Evitar DeadLock

VRE [5 3 2] VRD [3 2 0]

MRA
$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

MRR
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

MRPN
$$\begin{pmatrix} 2 & 2 & 1 \\ 3 & 2 & 2 \\ 1 & 2 & 2 \end{pmatrix}$$

Eventos:

P1 solicita 1R1, 1R2, 1R3

P2 solicita 1R1, 1R3

Evitar DeadLock

VRE [5 3 2] VRD [3 2 0]

MRA
$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

MRR
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 2 & 0 \end{pmatrix}$$

MRPN
$$\begin{pmatrix} 2 & 2 & 1 \\ 3 & 2 & 2 \\ 1 & 2 & 2 \end{pmatrix}$$

Eventos:

P1 solicita 1R1, 1R2, 1R3

P2 solicita 1R1, 1R3

P3 solicita 1R1, 2R2

Alocar R2 a P3 gera probabilidade de *deadlock* > 0

Deadlock

Impedir

Impedir Deadlock

- Condições necessárias para existir Deadlock:
  - Exclusão Mútua
  - Posse e Espera
  - Não Preempção
  - Espera Circular

  - Deadlock só pode existir se essas 4 condições aparecerem JUNTAS.
    - E se eliminarmos 1 delas?
      - Deadlock torna-se IMPOSSÍVEL de ocorrer.

Impedir Deadlock

- Eliminando condições necessárias para existir Deadlock:
  - Exclusão Mútua
    - Impressora
      - Fila de Impressão - Finita
      - Problema: e se a fila encher?
    - Retardar o problema

Impedir Deadlock

- Eliminando condições necessárias para existir Deadlock:
  - Posse e Espera
    - Requisitar todos os recursos possivelmente necessários num só passo e antecipadamente
      - Atomicidade: ou aloca todos ou nenhum
    - Problemas:
      - Pode alocar recursos que não vai usar
      - Pode demorar muito para usar um recursos alocado cedo demais
      - S.O. precisa manter quantidade alta de recursos do mesmo tipo para manter concorrência

Impedir Deadlock

- Exemplo
  - ***VRE = [10 8 6]***
  - Se P1 executar primeiro, P2 pode executar?
    - Não, pois não pode alocar 6 R1
  - Se P1 executar primeiro, P3 pode executar? Sim
  - Se P2 executar primeiro, outro pode executar? Não.

R1	R2	R3
5	3	3
6	5	2
4	4	3

MRR

Impedir Deadlock

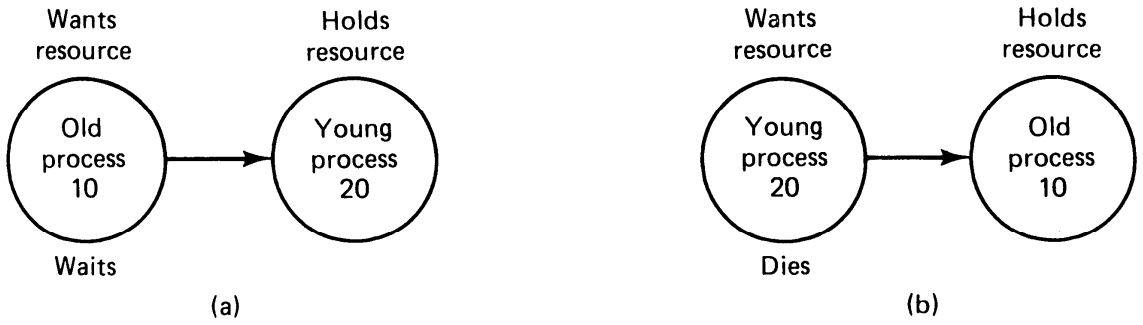
- Eliminando condições necessárias para existir Deadlock:
  - Eliminar a Espera Circular
    - Priorizar Recursos, numerando-os.
    - Requisitar recursos por ordem de prioridade

Impedir Deadlock

- Exemplo
  - ***VRE = [10 8 6]***
    - P1 requisita 4 R1
    - P1 requisita 1 R2
    - P1 só pode requisitar novos R2 e R3
    - P2 requisita 3R1 e 4 R2
    - P2 requisita 1 R3
    - P2 só pode requisitar novos R3
    - Elimina Espera Circular

Deadlock

- Prevenção
  - Ordenamento hierárquico de processos
  - Wait-Die (WD): Não-preemptivo
    - Quando *P* requer um recurso alocado por *Q* , *P* aguarda apenas se for mais velho que *Q*. Em outro caso, *P* morre.



Deadlock

- Prevenção
  - Wound-Wait (WW): Preemptivo
    - Quando *P* requer um recurso alocado por *Q* , *P* aguarda apenas se for mais novo que *Q*. Em outro caso, *Q* morre, liberando os recursos.