UFRJ – IM - DCC

Sistemas Operacionais I

Unidade II – Deadlock e Starvation

Organização da Unidade

- **Processos**
- **Threads**
- **Concorrência**
- ***Deadlock* e *Starvation***
  - Definições
  - Tipos de Recursos
  - Condições
  - Prevenção
  - Impedimento
  - Detecção
  - Recuperação

Definições

Deadlock

Bloqueio permanente de um conjunto de processos que competem por recursos ou se comunicam

Starvation

Situação em que um processo fica impossibilitado de ser executado por falta de chance de obter os recursos que necessita

exemplo:

processo R	processo S
obtém A	obtém B
requer B ⟺	requer A
•	•
•	•
•	•

Deadlock e Starvation

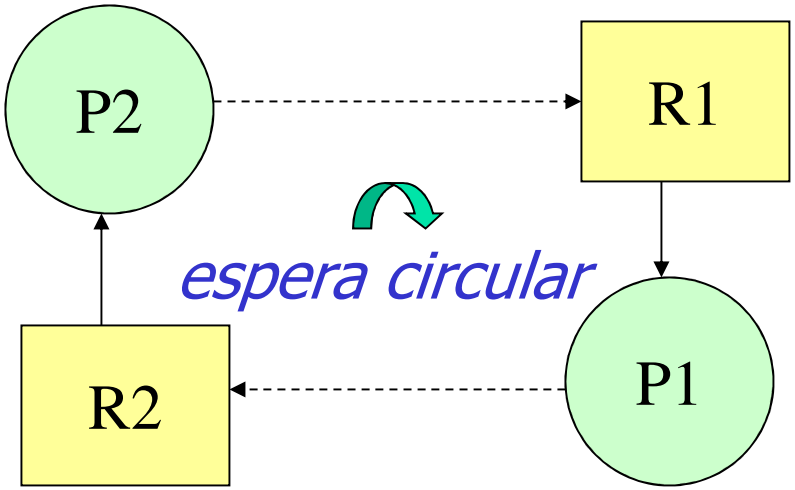
Deadlock ou Starvation?

Tipos de Recurso

- Reutilizáveis: continuam a existir após o uso
  - processador, interface de I/O, memória, arquivo
- Consumíveis: uma vez utilizados deixam de existir
  - mensagem, buffer de I/O

Condições para *Deadlock*

- Necessárias (porém não suficientes)
  - Exclusão mútua
  - Posse e espera (hold & wait)
  - Não preempção
- Suficiente
  - Espera circular



Estratégias para Tratamento de Deadlock

- Prevention - *Prevenir*
  - Objetivo: excluir a priori a possibilidade de ocorrência de Deadlocks.
- Avoidance - *Evitar*
  - Objetivo: relaxar as restrições do Prevention tomando medidas cautelosas para evitar a ocorrência de Deadlocks.
- Detection - *Detectar*
  - Objetivo: detectar a ocorrência e tomar providências para eliminar o Deadlock.

Prevention

Método Indireto :

Previnir a ocorrência de uma das 3 condições necessárias.

- Exclusão mútua: não pode ser eliminada.
- Posse e espera: processo pode alocar todos os recursos de uma só vez.
- Não preempção: um processo deve liberar os recursos obtidos caso outro recurso seja negado.

Prevention

Método direto :

Previne a ocorrência da espera circular definindo uma ordem (sequência) para requisição dos recursos

$$R_i \rightarrow R_j, \quad i<j$$

Exemplo: Processo A obtém R_i e solicita R_j,
Processo B não pode obter R_j e solicitar R_i

Avoidance

- Exige conhecimento de requisições futuras
- Duas abordagens
  - processo não é iniciado
  - alocação não é permitida

Abordagem 1: processo não é iniciado

- m recursos diferentes e n processos
- Requer conhecimento antecipado de todas as necessidades de recursos de um processo

$$\text{Recursos} = (r_1, r_2, r_3 \ldots r_m)$$ ***total existente***

$$\text{Disponibilidade} = (v_1, v_2, v_3 \ldots v_m)$$ ***total disponível***

Matriz de Requisições dos Processos Ativos

$$Claim = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1m} \\ & \cdots & & \\ & & \cdots & \\ c_{n1} & c_{n2} & \cdots & c_{nm} \end{pmatrix}$$

(*) c_{1k} – processo P_1 requer c cópias do recurso R_k

Matriz de Alocações de Recursos para os Processos

$$Allocation = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ & \cdots & & \\ & & \cdots & \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}$$

(*) a_{1k} – recurso R_k está alocado ao Processo P_1

Condições

1 - para todo i, todos os recursos estão disponíveis ou alocados

$$r_i = v_i + \sum_{k=1}^{n} a_{ki}$$

2 - para todo k,i , nenhum processo pode requerer mais que o total de recursos

$$c_{ki} \leq r_i \ \ para\ todo\ k,i$$

3 - para todo k,i , nenhum processo pode alocar mais recursos que a solicitação inicial

$$a_{ki} \leq c_{ki} \ \ para\ todo\ k,i$$

Regra de Decisão

Um novo processo só é iniciado se o máximo de requisições de todos os processos mais as requisições do novo processo possam ser atendidas

$$r_i \geq c_{(n+1)i} + \sum_{k=1}^{n} c_{ki}$$

Abordagem 2: alocação é recusada

Condição restritiva é relaxada

$$r_i \not\geq c_{(n+1)i} + \sum_{k=1}^{n} c_{ki}$$

A cada alocação é testado o Estado Seguro

Estado Seguro: existe uma sequência em que todos os processos executam até o final sem deadlock.

(*) faz uso das mesmas matrizes da abordagem 1

Exemplo:

R1	R2	R3
9	3	6

Recursos disponíveis

R1	R2	R3
0	1	1

Vetor de disponibilidade

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Matriz de Requisições

	R1	R2	R3
P1	1	0	0
P2	6	1	2
P3	2	1	1
P4	0	0	2

Matriz de Alocação

Estado Inicial

Exemplo:

R1	R2	R3
9	3	6

Recursos disponíveis

R1	R2	R3
6	2	3

Vetor de disponibilidade

	R1	R2	R3
P1	3	2	2
P2	0	0	0
P3	3	1	4
P4	4	2	2

Matriz de Requisições

	R1	R2	R3
P1	1	0	0
P2	0	0	0
P3	2	1	1
P4	0	0	2

Matriz de Alocação

P2 executa completamente

Exemplo:

R1	R2	R3
9	3	6

Recursos disponíveis

R1	R2	R3
9	2	4

Vetor de disponibilidade

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	3	1	4
P4	4	2	2

Matriz de Requisições

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	2	1	1
P4	0	0	2

Matriz de Alocação

P1 executa completamente

Exemplo:

R1	R2	R3
9	3	6

Recursos disponíveis

R1	R2	R3
9	3	4

Vetor de disponibilidade

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	4	2	2

Matriz de Requisições

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	2

Matriz de Alocação

P3 executa completamente

Exemplo:

R1	R2	R3
9	3	6

Recursos disponíveis

R1	R2	R3
9	3	6

Vetor de disponibilidade

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	0

Matriz de Requisições

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	0

Matriz de Alocação

P4 executa completamente

Outro Exemplo:

R1	R2	R3
9	3	6

Recursos disponíveis

R1	R2	R3
1	1	2

Vetor de disponibilidade

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Matriz de Requisições

	R1	R2	R3
P1	1	0	0
P2	5	1	1
P3	2	1	1
P4	0	0	2

Matriz de Alocação

Estado Inicial

Estado seguro????

Outro Exemplo:

R1	R2	R3
9	3	6

Recursos disponíveis

R1	R2	R3
0	1	1

Vetor de disponibilidade

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Matriz de Requisições

	R1	R2	R3
P1	2	0	1
P2	5	1	1
P3	2	1	1
P4	0	0	2

Matriz de Alocação

Suponha que P1 requisitou 1 unidade de R1 e 1 unidade de R3 e GANHOU!

Detecção

- verificação pode ser feita a cada requisição ou menos frequente
- matriz de requisição: q_{ij} representa a quantidade de recursos do tipo j requisitado por pelo processo i

- marcar cada processo que tem uma linha de "zeros" na matriz de alocação
- inicializar um vetor temporário w igual ao de disponibilidade
- identificar um índice i cujo processo não esteja marcado e a i-ésima linha de q_{ij} seja $\leq w_{k.}$
- se a linha não existir, terminar a sequência, caso contrário, marcar processo i somar a linha correspondente da matriz de alocação em w : $w_{k+1} = w_k + a_{ik}$

Existe um *deadlock* se existir um processo não marcado ao final do algoritmo

Exemplo:

R1	R2	R3	R4	R5
2	1	1	2	1

Recursos disponíveis

R1	R2	R3	R4	R5
0	0	0	0	1

Vetor de disponibilidade

	R1	R2	R3	R4	R5
P1	0	1	0	0	1
P2	0	0	1	0	1
P3	0	0	0	0	1
P4	1	0	1	0	1

Matriz de Requisições

	R1	R2	R3	R4	R5
P1	1	0	1	1	0
P2	1	1	0	0	0
P3	0	0	0	1	0
P4	0	0	0	0	0

Matriz de Alocação

Estado Inicial W = Vetor de disponibilidade = (0 0 0 0 1)

Exemplo:

R1	R2	R3	R4	R5
2	1	1	2	1

Recursos disponíveis

R1	R2	R3	R4	R5
0	0	0	0	1

Vetor de disponibilidade

	R1	R2	R3	R4	R5
P1	0	1	0	0	1
P2	0	0	1	0	1
P3	0	0	0	0	1
P4	1	0	1	0	1

Matriz de Requisições

	R1	R2	R3	R4	R5
P1	1	0	1	1	0
P2	1	1	0	0	0
P3	0	0	0	1	0
P4	0	0	0	0	0

*

Matriz de Alocação

1 – Marque P4, porque P4 não tem recursos alocados

Exemplo:

R1	R2	R3	R4	R5
2	1	1	2	1

Recursos disponíveis

R1	R2	R3	R4	R5
0	0	0	0	1

Vetor de disponibilidade

	R1	R2	R3	R4	R5
P1	0	1	0	0	1
P2	0	0	1	0	1
P3	0	0	0	0	1
P4	1	0	1	0	1

Matriz de Requisições

	R1	R2	R3	R4	R5	
P1	1	0	1	1	0	
P2	1	1	0	0	0	
P3	0	0	0	1	0	*
P4	0	0	0	0	0	*

Matriz de Alocação

W = W + (0 0 0 1 0)

W = (0 0 0 1 1)

2 – O pedido de P3 é menor ou igual a W, logo marque P3

Exemplo:

R1	R2	R3	R4	R5
2	1	1	2	1

Recursos disponíveis

R1	R2	R3	R4	R5
0	0	0	0	1

Vetor de disponibilidade

	R1	R2	R3	R4	R5
P1	0	1	0	0	1
P2	0	0	1	0	1
P3	0	0	0	0	1
P4	1	0	1	0	1

Matriz de Requisições

	R1	R2	R3	R4	R5	
P1	1	0	1	1	0	
P2	1	1	0	0	0	
P3	0	0	0	1	0	*
P4	0	0	0	0	0	*

Matriz de Alocação

Termina o algoritmo porque P1 e P2 não foram marcados **W = (0 0 0 1 1)**

Recuperação

1. Aborta todos os processos (mais usada)
2. Retorna processo a um checkpoint anterior e reinicia todos os processos
3. Aborta sucessivamente os processos
4. Retira sucessivamente os recursos

Escolha dos processos para os itens 3 e 4:

- menor tempo de processamento consumido
- menor quantidade de saídas produzidas
- maior tempo restante
- menos recursos alocados
- menor prioridade

Estratégias integradas

- Recursos divididos em classes
- Ordenamento entre classes para prevenir espera circular
- Dentro de uma classe, usar o mais apropriado para a classe

exemplo

Classes		Estratégia
área de swap	→	prevenção (hold & wait)
recursos	→	impedimento
memória	→	prevenção (preempção)
recursos internos	→	ordenação