Sistemas Operacionais

Aula 14 – Deadlock - Parte 01

Professor: Wellington Franco

Introdução

- Em ambientes de multiprogramação, vários processos podem competir por uma quantidade finita de recursos.

- Um processo solicita recursos, se os recursos não estão disponíveis naquele momento, o processo entra em estado de espera.

- Em alguns casos, um processo em espera não consegue mudar novamente de estado, porque os recursos que ele solicitou estão reservados para outros processos em espera.

Introdução

- Em ambientes de multiprogramação, vários processos podem competir por uma quantidade finita de recursos.
- Um processo solicita recursos, se os recursos não estão disponíveis naquele momento, o processo entra em estado de espera.
- Em alguns casos, um processo em espera não consegue mudar novamente de estado, porque os recursos que ele solicitou estão reservados para outros processos em espera.
- **DEADLOCK**

Definição de Deadlock

> *Um conjunto de processos está em estado de deadlock quando cada processo do conjunto está esperando por um evento que só pode ser causado por outro processo do conjunto. Os eventos que estamos mais interessados aqui são a aquisição e liberação de recursos. Os arquivos podem ser **físicos** ou **lógicos**,*
>
> **SILBERSCHARTZ, P. B. GALVIN, G. GAGNE (2009)**

Introdução

Modelo de sistema

- Basicamente, o sistema é baseado em recursos disponíveis para utilização pelos processos.

- Sob condições normais de operação, um processo só pode utilizar um recurso obedecendo a seguinte sequência:
  - **Solicitação**: o processo solicita o recurso. Se a solicitação não puder ser atendida imediatamente, então o processo solicitante deve esperar até poder adquirir o recurso.

  - **Uso**: o processo pode operar sobre o recurso.

  - **Liberação:** o processo libera o recurso.

Modelo de sistema

- Exemplo
  - O sistema possui dois discos rígidos

  - P0 e P1 mantém cada um o controle de um dos discos e deseja obter o controle do outro disco

Modelo de sistema

- Exemplo
  - Semáforos A e B, inicializados com 1

P_0	P_1
wait (A);	wait (B);
wait (B);	wait(A);

Caracterização do deadlock

- Em um deadlock, os processos nunca terminam sua execução e os recursos do sistema ficam ocupados, impedindo que outros jobs comecem a ser executados

- Uma situação de deadlock pode surgir nessas condições a seguir:

Caracterização do deadlock

- **Exclusão mútua:** apenas um processo de cada vez pode usar o recurso.
- **Posse e espera**: um processo em posse de um recurso pode esperar para adquirir recursos adicionais em posse de outros processos
- **Inexistência de preempção**: um recurso só pode ser liberado voluntariamente pelo processo que o está utilizando
- **Espera circular**: existe um conjunto de processos bloqueados $\{P_0, P_1, \ldots, P_n\}$ onde P0 aguarda por um recurso em posse de P1, P1 aguarda por um recurso em posse de P2, ..., Pn–1 aguarda por um recurso em posse de Pn, e Pn aguarda um recurso em posse de P0.

Grafo de alocação de recursos

- Os deadlocks podem ser descritos com mais precisão em um grafo orientado chamado de grafo de alocação de recursos do sistema.

- Grafo é composto de vértices V e arestas A

    - Os vértice V são compostos pelos

        - **processos** (P = {P1, P2, ..., Pn}); e

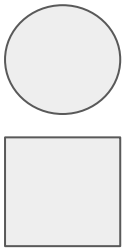
        - **recursos** (R = {R1, R2, ..., Rm})

Grafo de alocação de recursos

- Grafo é composto de vértices V e arestas A
    - As arestas A são as relações entre processos e recursos, diferenciados pela orientação da seta.
        - **Aresta de solicitação**: uma aresta de um processo para um recurso (P_i -> R_j)
        - **Aresta de atribuição**: uma aresta de um recurso para um processo (R_j -> P_i)

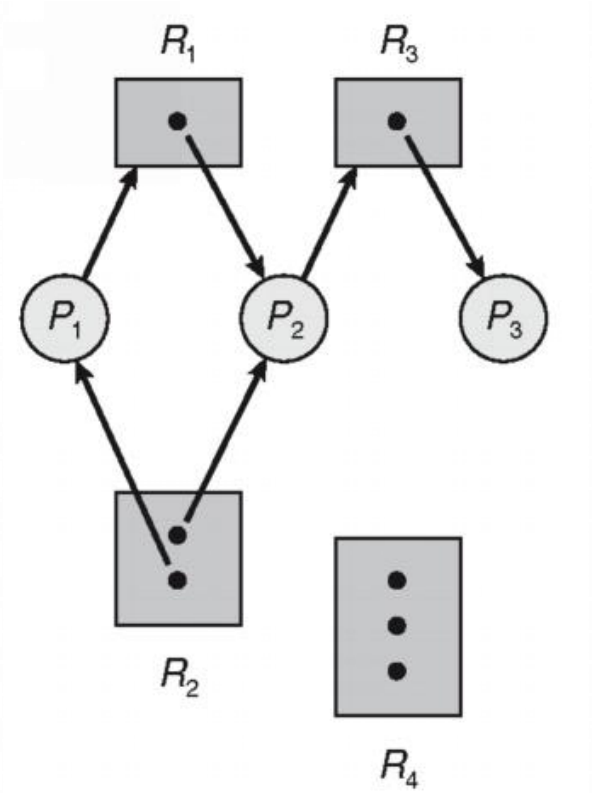
Grafo de alocação de recursos

- Exemplo:
  - P = P_1, P_2, P_3
  - R = R_1, R_2, R_3, R_4

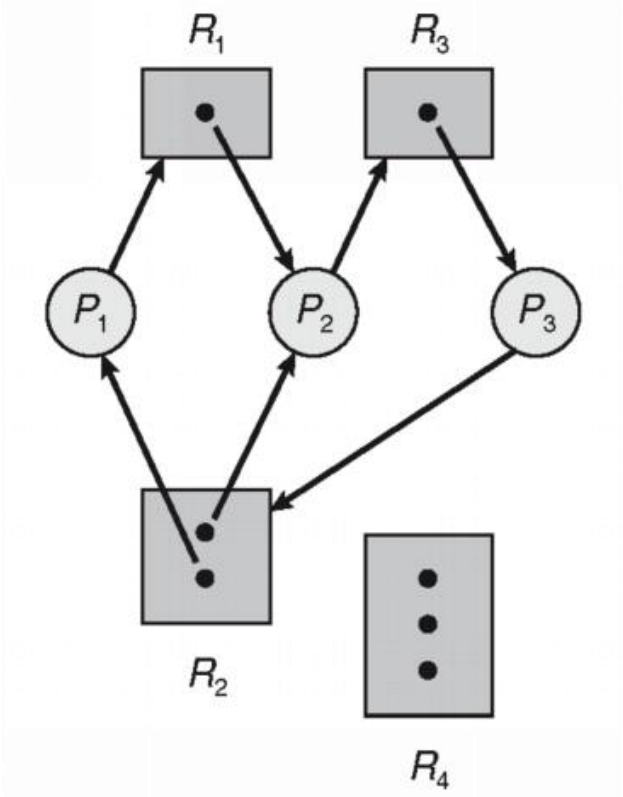


  - A = P_1-> R_1, P_2 -> R_3, R_1 -> P_2, R_2 -> P_2, R_2 -> P_1, R_3 -> P_3
  - Instâncias de recursos:
    - Uma instância do tipo de recurso R_1
    - Duas instâncias do tipo de recurso R_2
    - Uma instância do tipo de recurso R_3
    - Três instâncias do tipo de recurso R_4

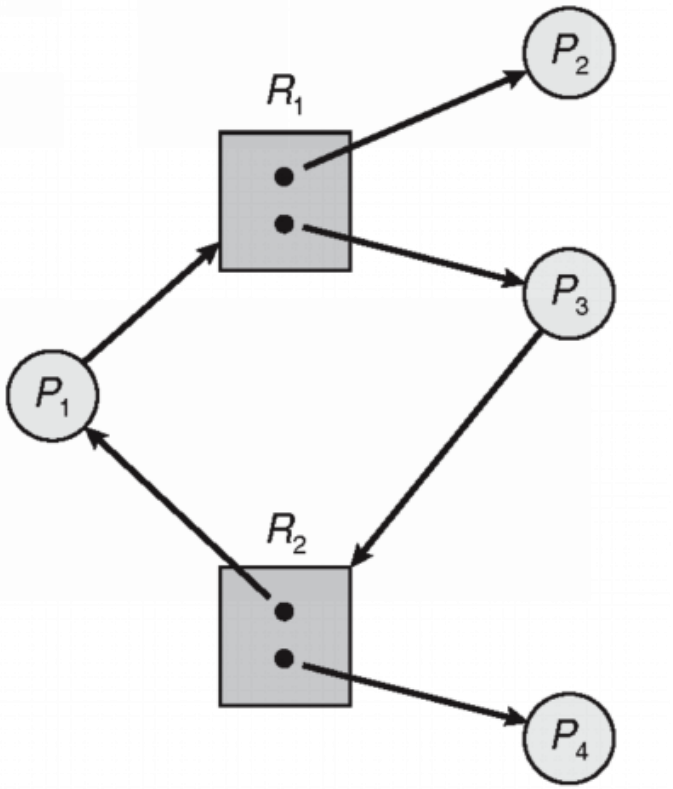
Exemplo 1 - Grafo de alocação de recursos



Exemplo 2 - Grafo de alocação de recursos



Exemplo 3 - Grafo de alocação de recursos



Métodos para manipulação de deadlocks

- Garantir que o sistema nunca entrará em um estado de deadlock
- Permitir que o sistema entre em deadlock e recuperar o sistema
- Ignorar o problema e fingir que deadlocks nunca ocorrem

Prevenção de Deadlocks

> *O desenvolvimento de uma ordenação não impede, por si só, o deadlock. É responsabilidade dos desenvolvedores de aplicações escrever programas que sigam a ordenação. Observe também que a distribuição dos identificadores deve seguir a ordem natural de uso dos recursos do sistema.*
>
> **SILBERSCHARTZ, P. B. GALVIN, G. GAGNE (2009)**

Prevenção de deadlock

- Para que um deadlock ocorra, cada uma das quatro condições necessárias deve estar presente.

- Ao assegurar que pelo menos uma dessas condições não possa acontecer, podemos nos prevenir contra a ocorrência de um deadlock.

Prevenção de deadlock

- **Exclusão mútua**: não é necessária para recursos compartilháveis; precisa valer para recursos não compartilháveis
  - Um arquivo de leitura (compartilhável).
  - Impressora (Não compartilhável)

- **Posse e espera:** precisa garantir que sempre que um processo requisite novos recursos ele não esteja em posse de nenhum outro recurso.
  - Exigir que um processo requisite e receba todos os recursos antes de iniciar sua execução, ou permitir que um processo só requisite recursos quando não tiver nenhum recurso

Prevenção de deadlock

- **Não-preempção:** Se um processo que está de posse de um recurso solicita um outro recurso que não pode ser alocado imediatamente para ele, todos os recursos em seu poder são imediatamente liberados
  - Os recursos liberados são adicionados à lista de recursos necessários para a execução do processo
  - O processo será reiniciado quando puder obter novamente todos os recursos que já tinha e também os novos recursos de que necessitava

- **Espera circular:** impor uma ordenação total nos tipos de recursos e exigir que todos os processo requisitem os recursos em ordem crescente de numeração

Impedimento de deadlock

- A forma mais simples e útil requer que cada processo declare a quantidade máxima de cada tipo de recurso disponível que ele pode precisar

- O algoritmo de impedimento de deadlocks examina dinamicamente o estado da alocação de recursos para garantir que jamais ocorra uma condição de espera circular

- O estado da alocação de recursos é definido pela quantidade de recursos **disponíveis** e **alocados** e pelas **demandas máximas** dos processos

Estado de segurança

- Um estado é seguro quando o sistema pode alocar recursos a cada processo em alguma ordem e continuar evitando um deadlock

- O sistema está em um estado seguro se existe uma sequência $<P_1, P_2, ..., P_n>$ com todos os processos do sistema na qual para cada P_i, os recursos que P_i ainda pode precisar podem ser alocados com os recursos disponíveis no momento + recursos em posse dos processos P_j, com $j < i$

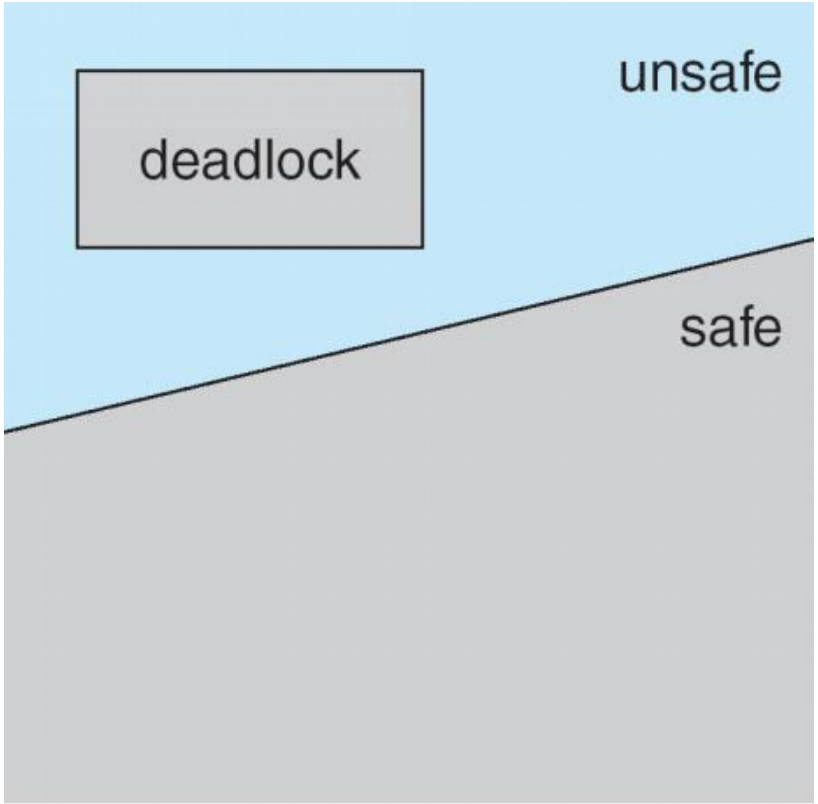
Estado de segurança

- Ou seja:
  - Se os recursos que P_i precisa não puderem ser fornecidos imediatamente, P_i pode esperar até que todos os processos P_j terminem
  - Quando P_j terminar, P_i pode obter os recursos necessários, executar, liberar os recursos e terminar
  - Quanto P_i termina, P_{i+1} pode obter os recursos de que necessita e assim por diante
  - Se tal situação não for possível, o **estado do sistema é inseguro**.

Estado de segurança

- Se o sistema está em um estado seguro, não há deadlock
- Se o sistema está em um estado inseguro, há possibilidade de deadlock
- **Impedimento:** garantir que o sistema jamais entre em um estado inseguro

Estado de segurança



Estado de segurança

- Exemplo
  - 12 drives de fita

Processos	Necessidades Máximas	Necessidades Correntes
P0	10	5
P1	4	2
P2	9	2

Estado seguro:
<P1, P0, P2>

Estado de segurança

- Exemplo
  - 12 drives de fita

Processos	Necessidades Máximas	Necessidades Correntes
P0	10	5
P1	4	2
P2	9	**3**

E se P2 receber mais um?

Algoritmos de impedimento de deadlocks

- **Instância única para cada recurso**
  - Utilizar um grafo de alocação de recursos
- **Múltiplas instâncias de algum tipo de recurso**
  - Utilizar o algoritmo do banqueiro

Algoritmo do grafo de alocação de recursos

- Utilizado quando se tem apenas uma instância de um cada recurso
- Além das arestas de solicitação e atribuição, tem a de **pretensão**
  - P_i -> R_j indica que o processo pode requerer um recurso em algum momento no futuro.
  - Simbolicamente, para diferenciar da solicitação, o traço da reta é **tracejado**

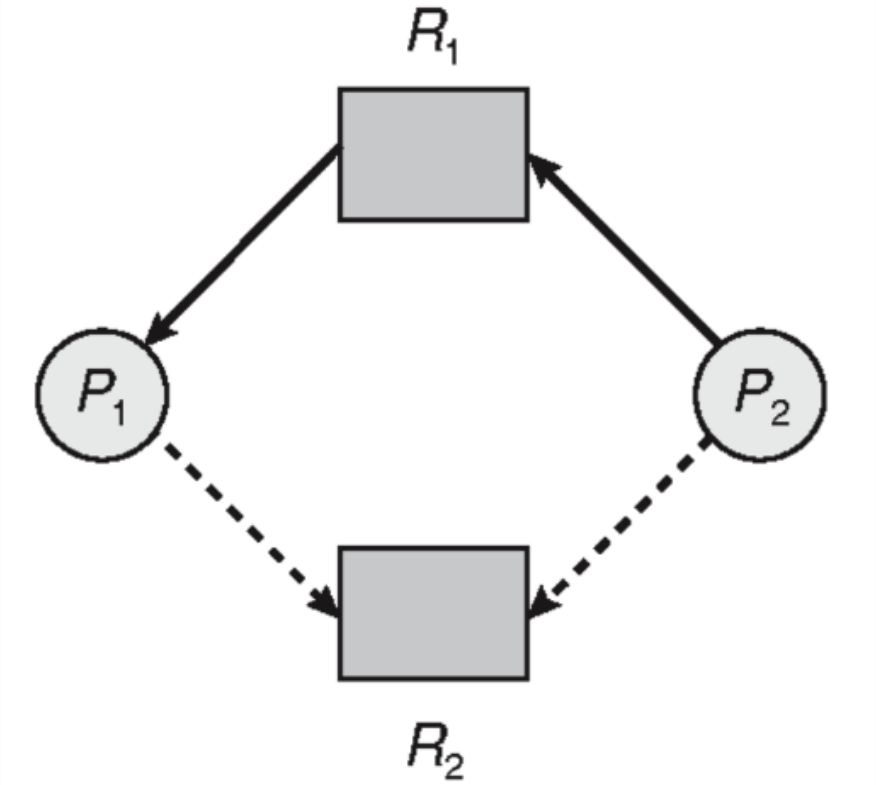
Algoritmo do grafo de alocação de recursos

- Se o processo solicitar o recurso, **a aresta de pretensão é convertida em solicitação**.

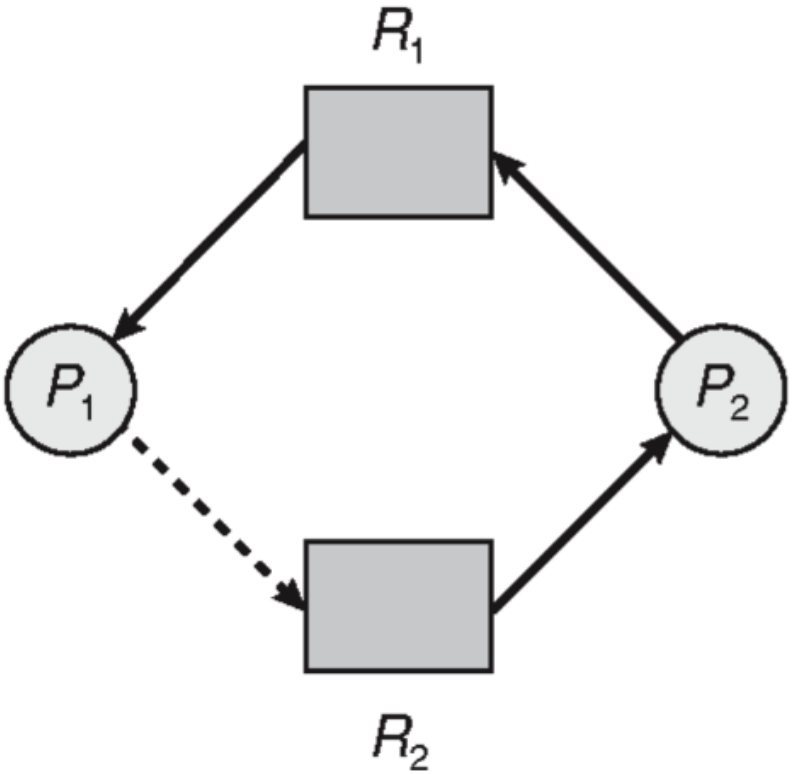
- Quando o recurso é liberado pelo processo, a **aresta de atribuição é convertida em aresta de pretensão**

- Os processos precisam anunciar a priori para o sistema quais recursos podem vir a solicitar, criando inicialmente todas as arestas de pretensão

Algoritmo do grafo de alocação de recursos



Algoritmo do grafo de alocação de recursos



Dúvidas??

E-mail: wellington@crateus.ufc.br