GSI018 – Sistemas Operacionais

Impasses (Deadlocks)

Universidade Federal de Uberlândia
Faculdade de Computação
Prof. Dr. rer. nat. Daniel D. Abdala

Na Aula Anterior...

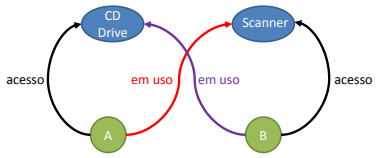
Nesta Aula

- Motivação acerca dos Impasses;
- Recursos e Seus Tipos;
- Ciclo de Alocação/Utilização de Recursos;
- Possibilidades de Espera;
- Modelagem de Impasses;
- Condições para Ocorrência de Impasses;
- Estratégias para Lidar com Impasses;
- Detecção de Impasses;
- Recuperação de Situações de Impasses;
- Evitando Empasses;
- Estados Seguros e Inseguros;

Impasse

- Muitos recursos em um Sistema Computacional são adequados ao uso de apenas um processo por vez;
- Todo SO deve ser capaz de garantir acesso exclusivo de um processo a certos recursos;
- O problema reside no fato de que processos geralmente requerem diversos recursos para executarem sua tarefa!
- Proc. A requer recursos ① e ② para executar;
- Proc. B também requer os mesmos recursos ① e ②;
- Proc. A tem acesso a ① e Processo B tem acesso a ②;
- Proc. A nunca ganha acesso a ② e similarmente Proc. B nunca ganha acesso a ① ;
- Formalmente: ***"Um conjunto de processos estará em situação de impasse se todo processo pertencente ao conjunto estiver esperando por um evento que somente outro processo deste mesmo conjunto poderá fazer acontecer."***

Impasse (Deadlock)



- Como detectar Impasses?
- Como resolver Impasses?

Recurso

- Dispositivo, arquivo, entrada em uma estrutura de dados, etc, que pode ser adquirido exclusivamente por um processo;
- Recursos podem ser classificados como:
  - Preemptivos;
  - Não preemptivos;

Recurso Preemptivo

- É aquele que pode ser retirado do processo que o detêm sem nenhum prejuízo;

Ex:

- Proc. A deseja imprimir um arquivo;
- Arquivo deve ser carregado na memória para pré-processamento;
- Proc. A requisita acesso a impressora (concedido);
- Proc. B utiliza um "bom pedaço" da memória;
- Proc. A requisita mais memória (negado);
- Páginas do Proc. B são mapeadas para o Disco;
- Proc. A requisita mais memória (concedido);

7

Recurso Não Preemptivo

- Se removido do processo que o detêm há uma forte possibilidade de falha na computação do processo;

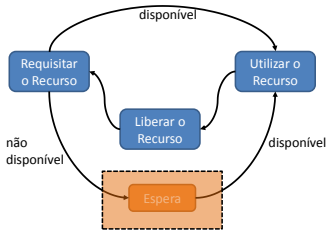
Ex: CD drive gravando um disco

- Em geral impasses envolvem recursos não preemptíveis;

8

Ciclo de Alocação/Utilização de Recursos

- É possível pensar abstratamente quanto ao ciclo de utilização de recursos da seguinte forma:



9

Possibilidades na Espera

- Duas Possibilidades:
  a) Processo permanece em um pequeno loop onde o recurso é requisitado continuamente;

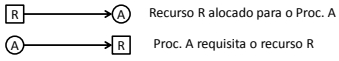
  Tecnicamente o processo nunca bloqueia, na prática, se o impasse não é resolvido, ocorrendo um loop infinito.

  a) O SO coloca o processo em um estado de bloqueio especial. O processo é "acordado" pelo SO quando o recurso se torna disponível;

10

Modelagem de Impasses

- Útil modelar formalmente impasses;
- Principalmente para detecção de impasses;
- Utilização de Grafos Dirigidos;
- Dois tipos de estados;
- Dois tipos de arestas;

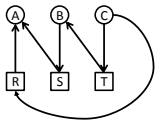


$$G = \{V, A\}$$
$$V = p_i \in P \vee r_i \in R$$
$$A = (p_i, p_j) / p_i \in P \wedge p_j \in R \vee p_i \in R \wedge p_j \in P$$

11

Modelagem de Impasses

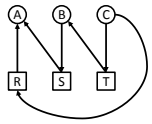
- Processos não requisitam a alocação de recursos todos aos mesmo tempo;
- **Ordem de alocação** – a ordem em que os recursos são requisitados pelo processo;
- Ex: Alloc={{A,R},{A,S},{B,S},{B,T},{C,T},{C,R}}



12

Modelagem de Impasses

- Processos não requisitam a alocação de recursos todos aos mesmo tempo;
- **Ordem de alocação** – a ordem em que os recursos são requisitados pelo processo;
- Ex: Alloc={{A,R},{A,S},{B,S},{B,T},{C,T},{C,R}}

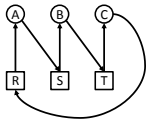


Não há impasse pois não há um ciclo!

13

Modelagem de Impasses

- Ex: Alloc={{A,R},{B,S},{C,T},{A,S},{B,T},{C,R})



Há um empasse!
Moral da história: a ordem de alocação importa!

14

Condições para Ocorrência de Impasses

1. **Exclusão Mútua**: Em um determinado instante, cada recurso está em uma de duas situações:
   a) Associado a um único processo;
   b) Disponível;
2. **Condição de Posse e Espera**: processos que em um determinado instante retêm recursos concedidos anteriormente podem requisitar novos recursos;
3. **Condição de Não Preempção**: recursos concedidos previamente a um processo não podem ser forçozamente tomados deste processo – eles devem ser explicitamente liberados pelo processo que os retém;
4. **Condição de Espera Circular**: Deve existir um encadeamento circular de dois ou mais processos. Cada um deles encontra-se à espera de um recurso que está sendo usado pelo membro seguinte da cadeia;

15

Estratégias Para Lidar Com Impasses

1. Simplesmente **ignorar** o problema;
2. **Detecção e Recuperação**: Deixar os impasses ocorrerem, detectá-los e corrigí-los;
3. **Anulação Dinâmica**: por meio de uma alocação cuidadosa de recursos;
4. **Prevenção**: Negando estruturalmente uma das quatro condições necessárias para gerar um impasse;

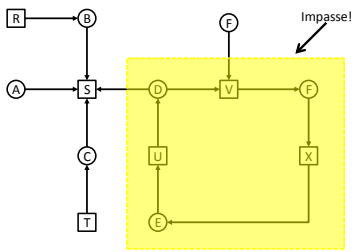
16

Detecção de Impasses

- Deixar o impasse ocorrer, detectá-lo e então tentar resolvê-lo:
- Em sua forma mais simples há apenas um recurso de cada tipo no sistema;
- Modelado via **grafo de recursos**;
- Qualquer processo que faça parte de um ciclo está em situação de impasse;

17

Exemplo



18

Detecção de Impasses

- Algoritmo para detecção de ciclos em grafos dirigidos;
- Estrutura de dados adequada para representar o grafo dirigido;
  - Estrutura alocada estática ou dinamicamente?
  - Custo em percorrer e atualizar a ED?
- Moral: **Detectar impasses é caro! Sempre vale a pela?**
- O que muda se houverem múltiplas instâncias de um mesmo recurso?

19

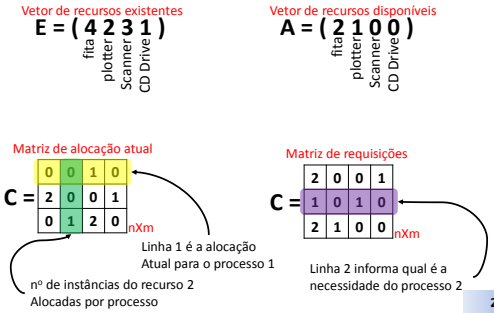
Detecção com Múltiplas Instâncias do Mesmo Recurso

- A representação em grafos direcionados não é mais suficiente;
- Possível Solução. Seja:
  - P_1, ...P_n → n Processos;
  - m → número de classes de recursos;
  - E_i, (1≤i≤m) → E é o vetor de recursos existentes. Fornece o número total de recursos existentes.
- Em um instante qualquer, alguns recursos se encontram alocados a processos e, por isso, não estão disponíveis;
  - A ⊆ E → A é o vetor de recursos disponíveis;
  - C → matriz de alocação atual;
  - R → matriz de requisição;

$$\sum_{i=1}^{n} C_{i,j} + A_j = E_j$$

20

Exemplo



21

Algoritmo de Detecção de Impasses

① Procure um processo desmarcado Pi, para o qual a i-ésima linha de R seja menor ou igual à correspondente de A;

② Se este processo for encontrado, adicione a i-ésima linha C a correspondente de A, marque o processo e volte para o passo 1;

③ Se não existir esse processo o algoritmo termina;

- Passo 1 Procura por um processo que possa ser executado até ser concluído;

22

Recuperação de Situações de Impasses

- Passos a serem seguidos uma vez que um impasse é detectado:
  - ① Recuperação por meio de preempção;
  - ② Recuperação por meio de retrocesso (checkpoints);
  - ③ Recuperação por meio de eliminação de processos;

23

Evitando Impasses

- SO deve ser capaz de decidir se liberar um recurso é seguro ou não;
- SO só deve alocar um recurso quando a alocação for segura;
- É possível evitar impasses?
  - Sim, porém há restrições;
  - Apenas é possível quando certas informações estiverem disponíveis antecipadamente;
- Algoritmos para evitar impasses – baseados no conceito de **estados seguros**;

24

Estados Seguros e Inseguros

- Um estado é considerado seguro se ele não está em situação de impasse e se existe alguma ordem de escalonamento na qual todo o processo possa ser executado até sua conclusão, mesmo se, repentinamente, todos eles requisitem, de uma só vez, o máximo possível de recursos;

25

Exemplo

	possui	max.
A	3	9
B	2	4
C	2	7

disponível = 3

	possui	max.
A	3	9
B	4	4
C	2	7

disponível = 1

	possui	max.
A	3	9
B	0	-
C	2	7

disponível = 5

	possui	max.
A	3	9
B	0	-
C	7	7

disponível = 0

	possui	max.
A	3	9
B	0	-
C	0	-

disponível = 7

	possui	max.
A	9	9
B	0	-
C	0	-

disponível = 1

	possui	max.
A	0	-
B	0	-
C	0	-

disponível = 10

26

Exemplo

	possui	max.
A	3	9
B	2	4
C	2	7

disponível = 3

	possui	max.
A	4	9
B	2	4
C	2	7

disponível = 2

	possui	max.
A	4	9
B	4	4
C	2	7

disponível = 0

	possui	max.
A	4	9
B	0	-
C	2	7

disponível = 4

Alocação leva a um estado não seguro

- Estado Inseguro ≠ Situação de Impasse!

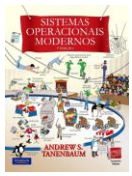
27

Algoritmo do Banqueiro para um Único Recurso

- Dijkstra, 1965
- Extensão do algoritmo de detecção de impasses;

28

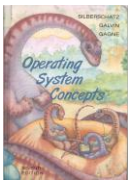
Bibliografia - Básica



- 3ª Edição
- Páginas 269 – 288

29

Bibliografia - Básica



- 7ª edição
- Páginas 245 – 271

30

Bibliografia - Adicional

- Capítulo 12
- Página 280