Sistemas Operacionais

Aula 16 – Deadlock

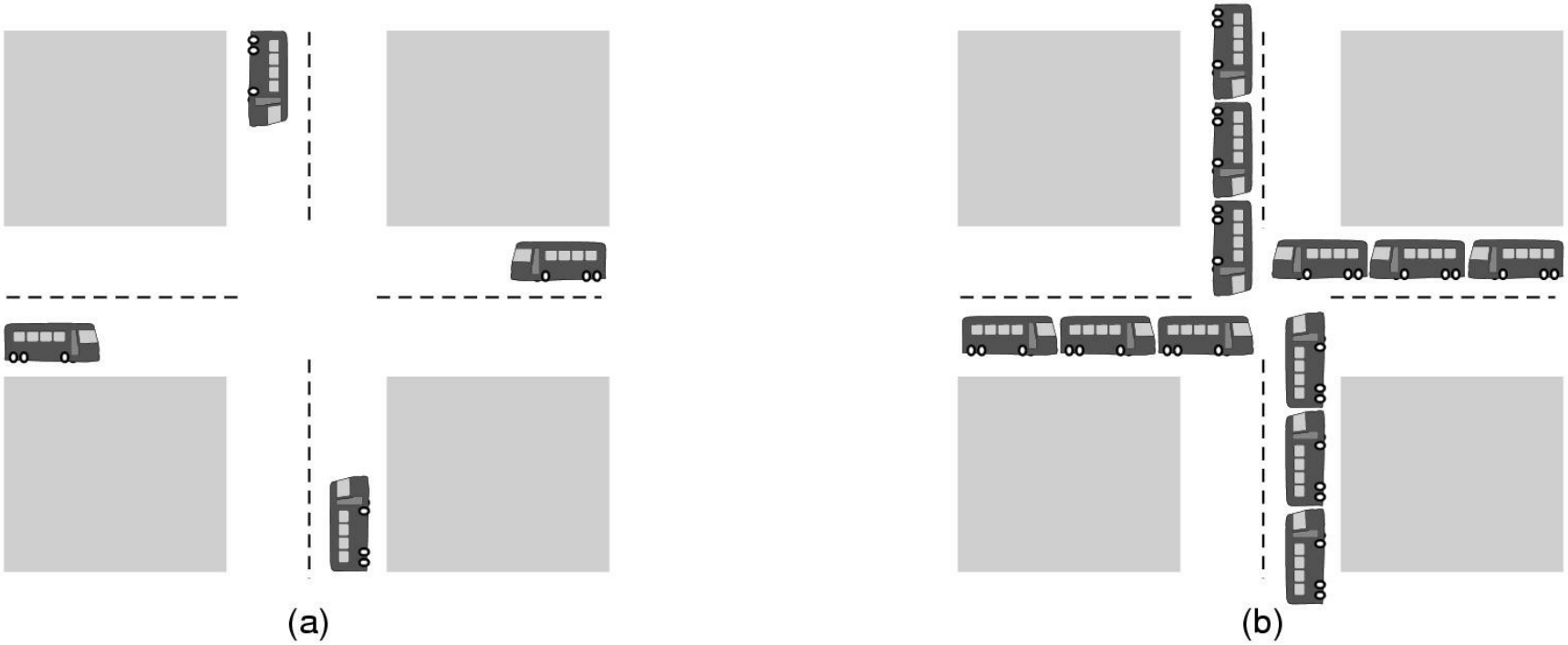
Prof. Igor da Penha Natal

Material baseado nos materiais de aula do prof. Jó Ueyama.

Deadlocks

- Dispositivos e recursos são compartilhados a todo momento: impressora, disco, arquivos, entre outros...;
- *Deadlock:* processos ficam parados sem possibilidade de poderem continuar seu processamento;

Deadlocks



Uma situação de *deadlock*


11455

Deadlocks

- Recursos:
  - Preemptivos: podem ser retirados do processo sem prejuízos;
    - Memória;
    - CPU;
  - Não-preemptivos: não podem ser retirados do processo, pois causam prejuízos;
    - CD-ROM;
    - Unidades de fita;
    - *Deadlocks* ocorrem com esse tipo de recurso;

Deadlocks

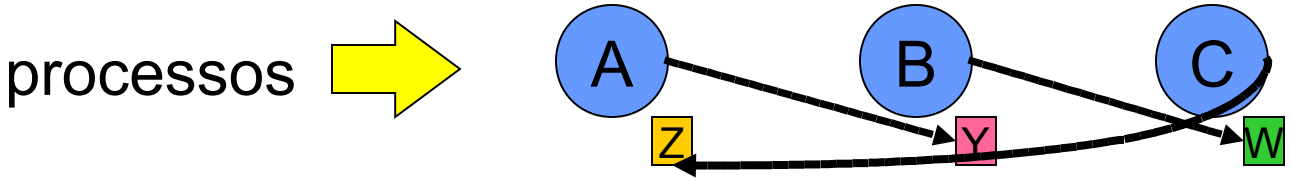
- Requisição de recursos/dispositivos:
  - Requisição do recurso;
  - Utilização do recurso;
  - Liberação do recurso;
- Se o recurso requerido não está disponível, duas situações podem ocorrer:
  - Processo que requisitou o recurso fica bloqueado até que o recurso seja liberado, ou;
  - Processo que requisitou o recurso falha, e depois de um certo tempo tenta novamente requisitar o recurso;

Deadlocks

- Quatro condições devem ocorrer para que um *deadlock* exista:
  - Exclusão mútua: um recurso só pode estar alocado para um processo em um determinado momento;
  - Uso e espera (*hold and wait*): processos que já possuem algum recurso podem requerer outros recursos;
  - Não-preempção: recursos já alocados não podem ser retirados do processo que os alocou; somente o processo que alocou os recursos pode liberá-los;
  - Espera Circular: um processo pode esperar por recursos alocados a outro processo;

Deadlocks

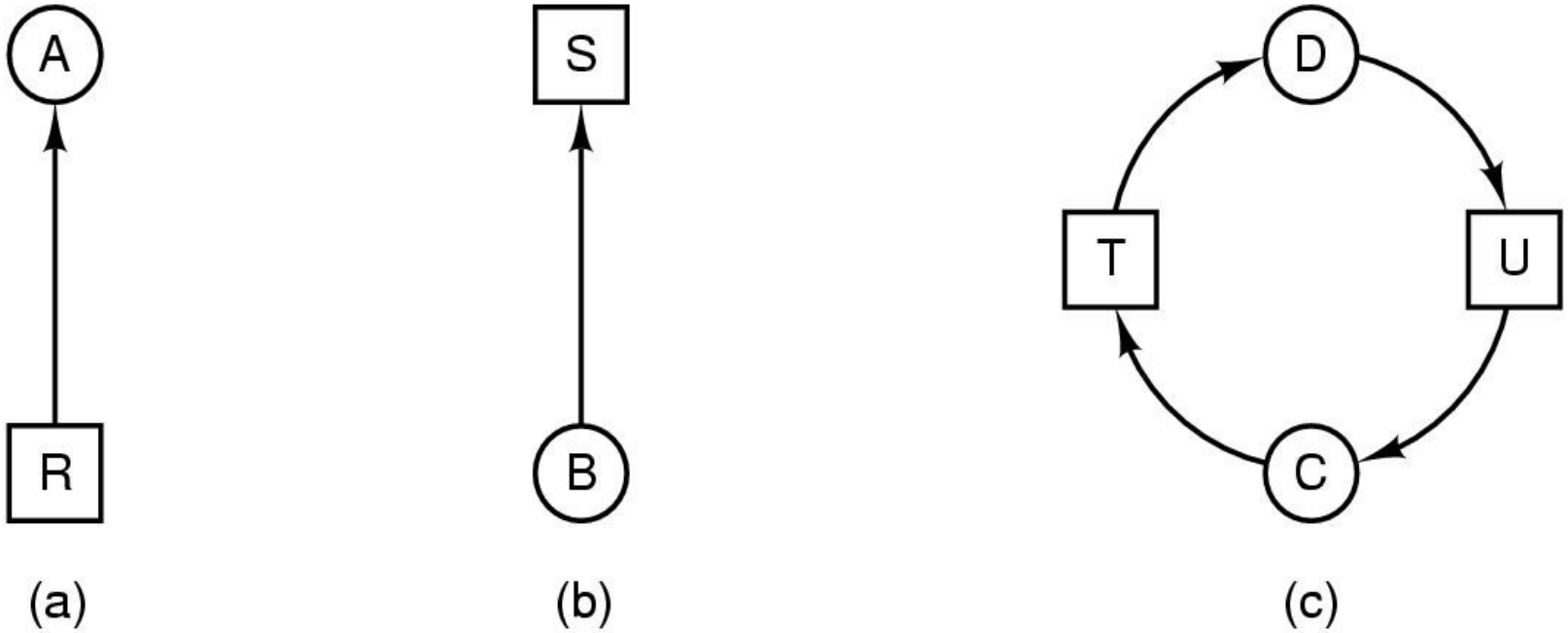
- Espera circular por recursos.
- Exemplo:
  - O processo “A” espera pelo processo “B”, que espera pelo processo “C”, que espera pelo processo “A”.



Deadlocks

- Geralmente, *deadlocks* são representados por grafos a fim de facilitar sua detecção, prevenção e recuperação
  - Ocorrência de ciclos pode levar a um *deadlock*;

Deadlocks
Grafos de alocação de recursos



a) **Recurso R** alocado ao **Processo A**
b) **Processo B** requisita **Recurso S**
c) ***Deadlock***

Deadlocks

- Quatro estratégias para tratar *deadlocks:*
  - Ignorar o problema;
  - Detectar e recuperar o problema;
  - Evitar dinamicamente o problema – alocação cuidadosa de recursos;
  - Prevenir o problema por meio da não satisfação de uma das quatro condições citadas anteriormente;

Tratamento de Deadlocks

- Ignorar o problema.
  - Comparar a freqüência de ocorrência de *deadlocks* com a freqüência de outras falhas do sistema.
    - Falhas de *hardware*, erros de compiladores, erros do Sistema Operacional, etc.
  - Se o esforço em solucionar o problema for muito grande em relação a freqüência com que o *deadlock* ocorre, ele pode ser ignorado.

Deadlocks

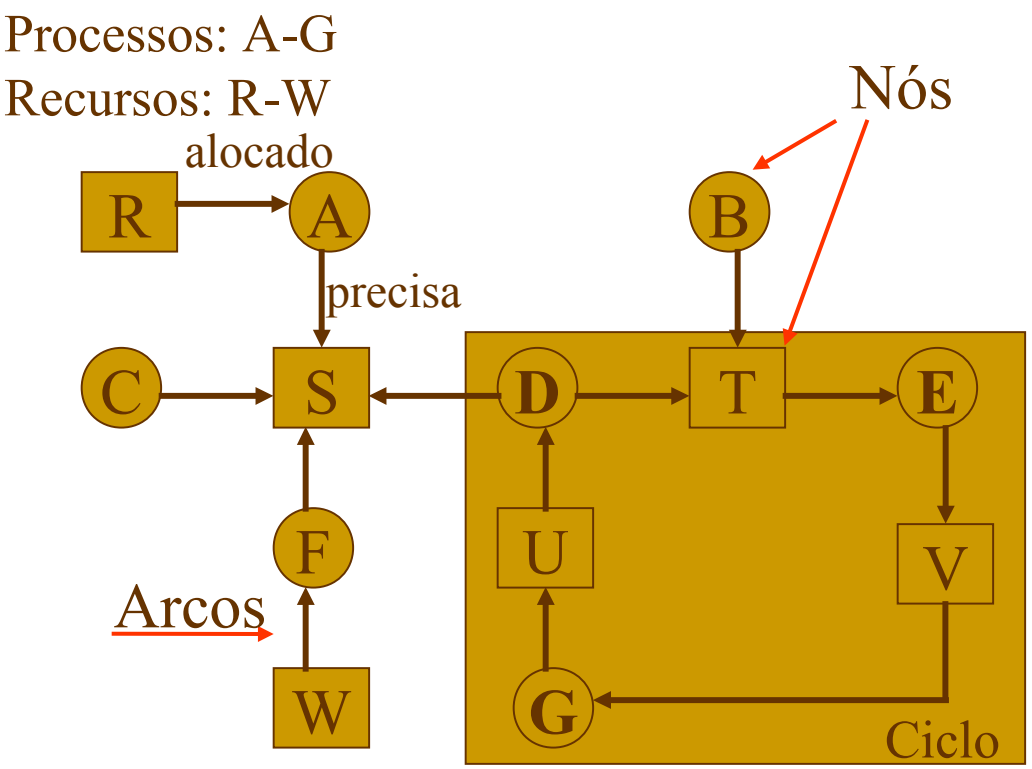
- Ignorar o problema:
  - Freqüência do problema;
  - Alto custo – estabelecimento de condições para o uso de recursos;
  - Algoritmo do AVESTRUZ;

Deadlocks

- Detectar e Recuperar o problema:
  - Processos estão com todos os recursos alocados;
  - Procedimento: Permite que os *deadlocks* ocorram, tenta detectar as causas e solucionar a situação;
  - Algoritmos:
    - Detecção com um recurso de cada tipo;
    - Detecção com vários recursos de cada tipo;
    - Recuperação por meio de preempção;
    - Recuperação por meio de *rollback* (volta ao passado);
    - Recuperação por meio de eliminação de processos;

Deadlocks

- Detecção com um recurso de cada tipo:
  - Construção de um grafo;
  - Se houver ciclos, existem potenciais *deadlocks*;



Situação:
PA usa R e precisa de S;
PB precisa de T;
PC precisa de S;
PD usa U e precisa de S e T;
PE usa T e precisa de V;
PF usa W e precisa de S;
PG usa V e precisa de U;

Pergunta:
Há possibilidade de *deadlock*?

Deadlocks

- Recuperação de *Deadlocks*:
  - Por meio de preempção: possibilidade de retirar temporariamente um recurso de seu atual dono (processo) e entregá-lo a outro processo;
  - Por meio de *rollback*: recursos alocados a um processo são armazenados em arquivos de verificação (checkpoint files); quando ocorre um *deadlock*, os processos voltam ao estado no qual estavam antes do *deadlock* → **solução cara**;

Deadlocks

- Recuperação de *Deadlocks*:
  - Por meio de eliminação de processos: processos que estão no ciclo com *deadlock* são retirados do ciclo;
  - Melhor solução para processos que não causam algum efeito negativo ao sistema;
    - Ex1.: compilação – sem problemas;
    - Ex2.: atualização de um base de dados – problemas;

Deadlocks

- **Evitar dinamicamente o problema**:
  - Alocação individual de recursos → à medida que o processo necessita;
  - Soluções também utilizam matrizes;
  - Escalonamento cuidadoso → alto custo;
    - Conhecimento prévio dos recursos que serão utilizados;
  - Algoritmos:
    - Banqueiro para um único tipo de recurso;
    - Banqueiro para vários tipos de recursos;
  - Definição de Estados Seguros e Inseguros;

Deadlocks

- Estados seguros: não provocam *deadlocks* e há uma maneira de atender a todas as requisições pendentes finalizando normalmente todos os processos;
  - A partir de um estado seguro, existe a garantia de que os processos terminarão;

- Estados inseguros: podem provocar *deadlocks,* mas não necessariamente provocam;
  - A partir de um estado inseguro, não é possível garantir que os processos terminarão corretamente;

Deadlocks

- Algoritmos do Banqueiro:
  - Idealizado por Dijkstra (1965);
  - Considera cada requisição no momento em que ela ocorre, verificando se essa requisição leva a um estado seguro; Se sim, a requisição é atendida, se não o atendimento é adiado para um outro momento;
  - Premissas adotadas por um banqueiro (SO) para garantir ou não crédito (recursos) para seus clientes (processos);
  - Nem todos os clientes (processos) precisam de toda a linha de crédito (recursos) disponível para eles;

Deadlocks

- Algoritmo do Banqueiro para um único tipo de recurso:

- Solicitações de crédito são realizadas de tempo em tempo;
- * C é atendido e libera 4 créditos, que podem ser usados por B ou D;

Deadlocks

- Algoritmo do Banqueiro para um único tipo de recurso:

	Possui	Máximo
A	0	6
B	0	5
C	0	4
D	0	7

Livre: 10

Seguro

	Possui	Máximo
A	1	6
B	1	5
C	2	4
D	4	7

Livre: 2

Seguro

	Possui	Máximo
A	1	6
B*	2	5
C	2	4
D	4	7

Livre: 1

Inseguro

- Solicitações de crédito são realizadas de tempo em tempo;
- * B é atendido. Em seguida os outros fazem solicitação, ninguém poderia ser atendido;

Deadlocks

- Algoritmo do Banqueiro para vários tipos de recursos:
  - Mesma idéia, mas duas matrizes são utilizadas;

Recursos → E = (6 3 4 2);
Alocados → P = (5 3 2 2);
Disponíveis → A = (1 0 2 0);

Processos	Unidade de Fita	Plotters	Impressoras	Unidade de CD-ROM
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

C = Recursos Alocados

A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

R = Recursos ainda necessários

Deadlocks

- Algoritmo do Banqueiro para vários tipos de recursos:

Processos	Unidade de Fita	*Plotters*	Impressoras	Unidade de CD-ROM
A	3	0	1	1
B	0	1	1	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

C = Recursos Alocados

Alocados → P = (5 3 **3** 2);
Disponíveis → A = (1 0 **1** 0);

A	1	1	0	0
B	0	1	0	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

R = Recursos ainda necessários

- Podem ser atendidos: D, A ou E, C;

Deadlocks

- Algoritmo do Banqueiro para vários tipos de recursos:

Processos	Unidade de Fita	Plotters	Impressoras	Unidade de CD-ROM
A	3	0	1	1
B	0	1	1	0
C	1	1	1	0
D	1	1	0	1
E	0	0	1	0

C = Recursos Alocados

Alocados → P = (5 3 **4** 2);
Disponíveis → A = (1 0 **0** 0);

A	1	1	0	0
B	0	1	0	2
C	3	1	0	0
D	0	0	1	0
E	2	1	0	0

R = Recursos ainda necessários

- *Deadlock → atender o processo E; Solução:*

Deadlocks

- Algoritmo do Banqueiro:
  - Desvantagens
    - Pouco utilizado, pois é difícil saber quais recursos serão necessários;
    - Escalonamento cuidadoso é caro para o sistema;
    - O número de processos é dinâmico e pode variar constantemente, tornando o algoritmo custoso;
  - Vantagem
    - Na teoria o algoritmo é ótimo;

Deadlocks

- Prevenir *Deadlocks*:
  - Atacar uma das quatro condições:

Condição	Abordagem
Exclusão Mútua	Alocar todos os recursos usando um *spool*
Uso e Espera	Requisitar todos os recursos inicialmente para execução – difícil saber; sobrecarga do sistema
Não-preempção	Retirar recursos dos processos – pode ser ruim dependendo do tipo de recurso; praticamente não implementável
Espera Circular	Ordenar numericamente os recursos, e realizar solicitações em ordem numérica Permitir que o processo utilize apenas um recurso por vez