Sistemas Operacionais

Deadlocks

Profª Drª Thaína Aparecida Azevedo Tosta

tosta.thaina@unifesp.br

UNIFESP
Universidade Federal de São Paulo

São José dos Campos
2024/01

Aula passada

- Introdução ao escalonamento de processos
- Escalonamento em sistemas em lote
- Escalonamento em sistemas interativos
- Escalonamento em sistemas de tempo real
- Política *vs* mecanismo de escalonamento
- Escalonamento de threads

Sumário

- Conceitos iniciais
- Recursos
- Introdução aos impasses
- Algoritmo do avestruz

Objetivo: questões de fixação.

Conceitos iniciais

- Os sistemas computacionais têm recursos que podem ser usados somente por um processo de cada vez, o que faz com que os sistemas operacionais tenham a capacidade de conceder (temporariamente) acesso exclusivo a um processo a determinados recursos;
- Exemplos (software e hardware):
  - Processos A (pede primeiro escâner) e B (pede primeiro gravador) querem gravar um documento escaneado em um disco Blu-ray;
  - Banco de dados com processos A e B que bloqueiam registros que estão usando.
- O que queremos? Prevenir ou evitar impasses (deadlocks).

Recursos

- Uma classe importante de impasses envolve recursos para os quais algum processo teve acesso exclusivo concedido;

- Recursos? Qualquer coisa que precisa ser adquirida, usada e liberada com o passar do tempo:

  Dispositivos de hardware (ex. unidade de Blu-ray);

  Registros de dados (em um banco);

  Arquivos.

Recursos

- Um **recurso preemptível** é aquele que pode ser retirado do processo proprietário sem causar-lhe prejuízo algum;
  - Sistema com uma memória de usuário de 1 GB, uma impressora e dois processos de 1 GB cada, onde ambos querem imprimir.
- Um **recurso não preemptível**, por sua vez, é um recurso que não pode ser tomado do seu proprietário atual sem potencialmente causar uma falha.
  - Processo executado em um Blu-ray.

Recursos

- O recurso é preemptível ou não? DEPENDE!
  - Memória em PC padrão é preemptível, mas não é em smartphones.
- Vamos focar em recursos não preemptíveis:
  1. Solicitar o recurso.
  2. Usar o recurso.
  3. Liberar o recurso.
- Tentou solicitar e não deu certo: tem que esperar (bloqueado, erro ou insistindo).

Recursos

Para alguns tipos de recursos, como registros em um sistema de banco de dados, cabe aos processos do usuário, em vez do sistema, gerenciar eles mesmos o uso de recursos, como por semáforos ou mutex.

```
typedef int semaphore;
semaphore resource_1;

void process_A(void) {
        down(&resource_1);
        use_resource_1( );
        up(&resource_1);
}
```

(a)

```
typedef int semaphore;
semaphore resource_1;
semaphore resource_2;

void process_A(void) {
        down(&resource_1);
        down(&resource_2);
        use_both_resources( );
        up(&resource_2);
        up(&resource_1);
}
```

(b)

Recursos

Com um único processo é fácil, e com dois?

(a) Solicitação na mesma ordem

(b) Solicitação em ordem inversa

(a) Código livre de impasses. (b) Código com impasse potencial.

```
typedef int semaphore;
semaphore resource_1;
semaphore resource_2;

void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources( );
    up(&resource_2);
    up(&resource_1);
}

void process_B(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources( );
    up(&resource_2);
    up(&resource_1);
}
```

(a)

```
semaphore resource_1;
semaphore resource_2;

void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources( );
    up(&resource_2);
    up(&resource_1);
}

void process_B(void) {
    down(&resource_2);
    down(&resource_1);
    use_both_resources( );
    up(&resource_1);
    up(&resource_2);
}
```

(b)

Introdução aos impasses

- Um conjunto de processos estará em situação de impasse se cada processo no conjunto estiver esperando por um evento que apenas outro processo no conjunto pode causar;
- Como todos os processos estão esperando, nenhum deles jamais causará qualquer evento que possa despertar outros, e todos os processos continuam a esperar para sempre;
- Nenhum dos processos pode executar, pode liberar quaisquer recursos e nenhum pode ser desperto: **impasse de recurso** (tipo mais comum).

Introdução aos impasses

- Quatro condições **têm de ser válidas** para haver um impasse (de recurso):

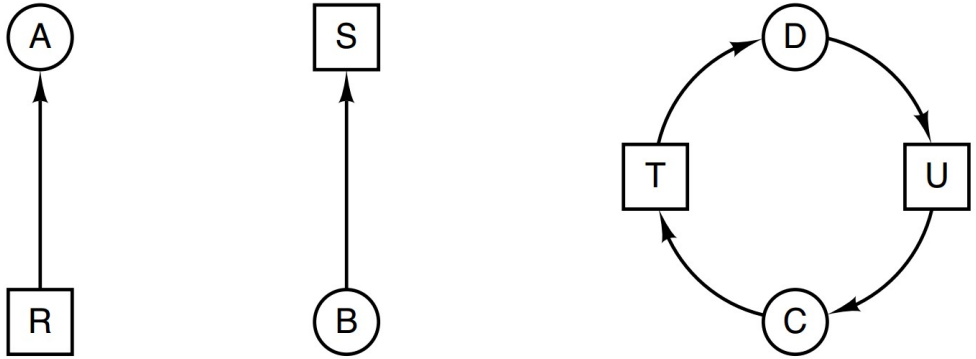
  1. Condição de exclusão mútua (do recurso para os processos);
  2. Condição de posse e espera (processos que solicitam novos recursos);
  3. Condição de não preempção (recursos concedidos não podem ser tomados à força);
  4. Condição de espera circular (lista de dois ou mais processos).

- Relação com a política que um sistema pode ter ou não.

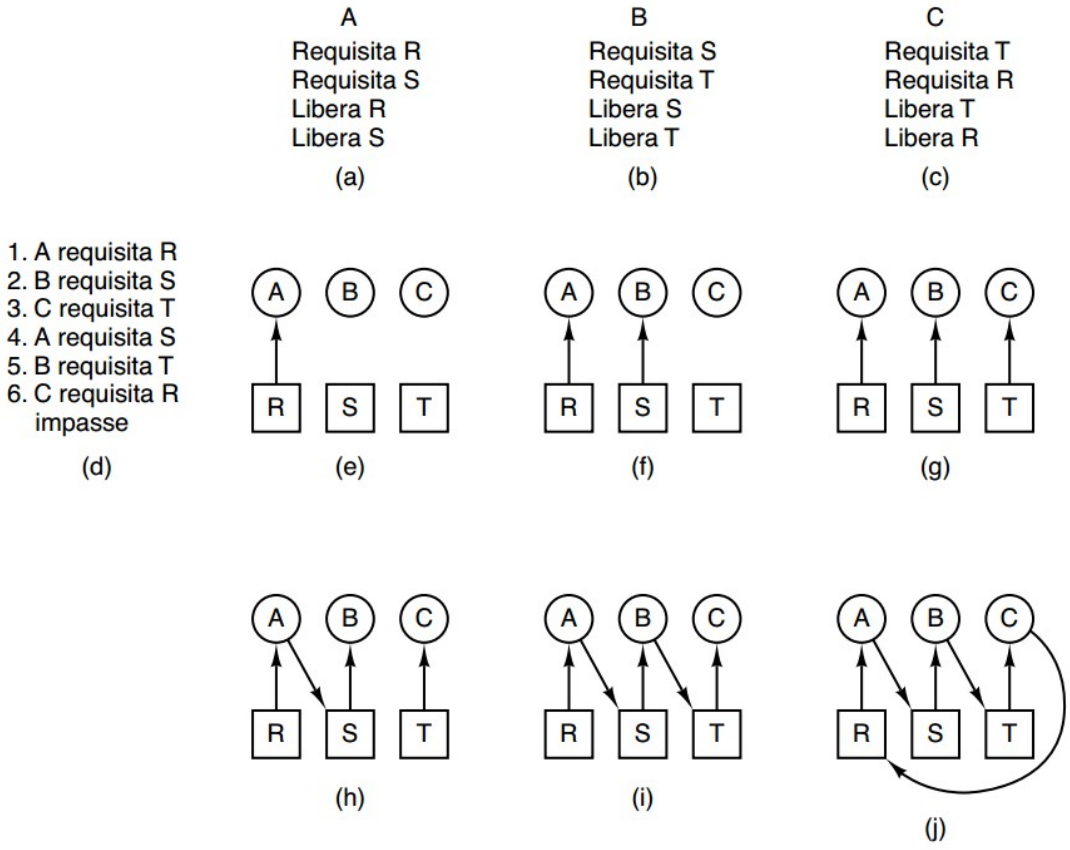
Introdução aos impasses

Holt (1972) demonstrou como essas quatro condições podem ser modeladas com grafos dirigidos (processos são círculos e recursos são quadrados).

Recurso R previamente solicitado, concedido e atualmente com o processo A.	O processo B está bloqueado esperando pelo recurso S.	Impasse com ciclo C-T-D-U-C.



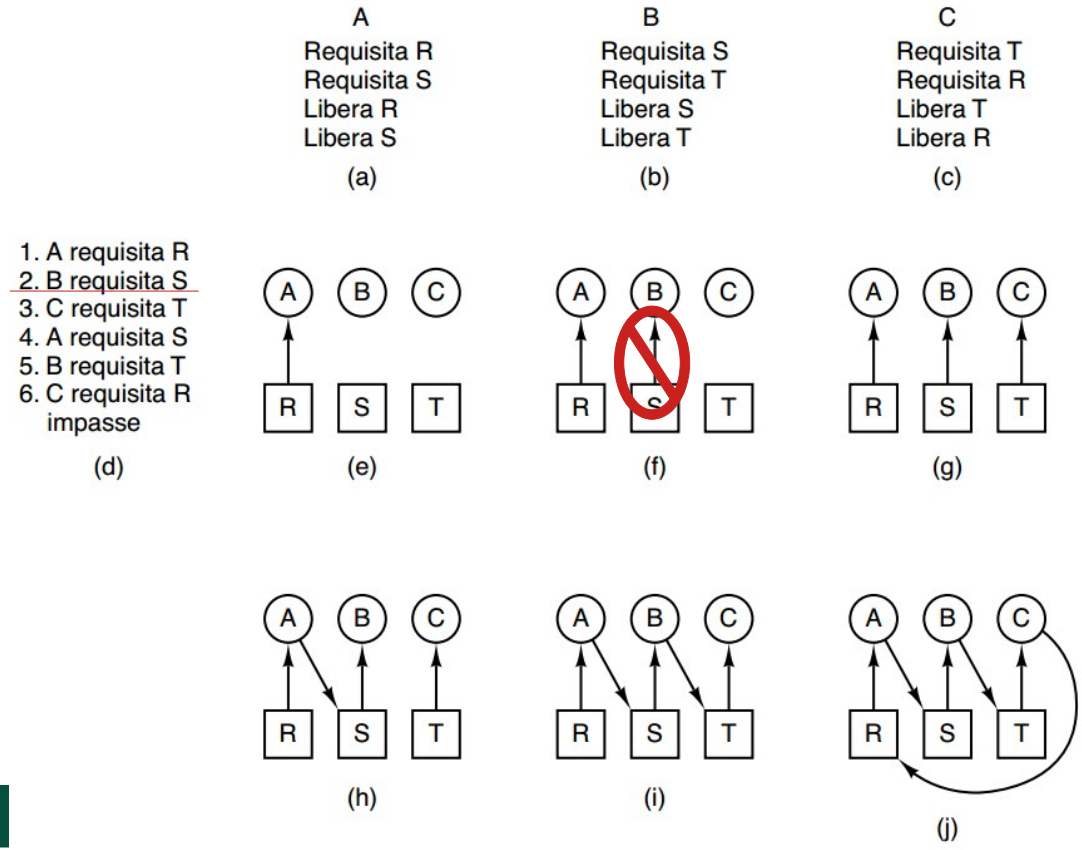
Introdução aos impasses



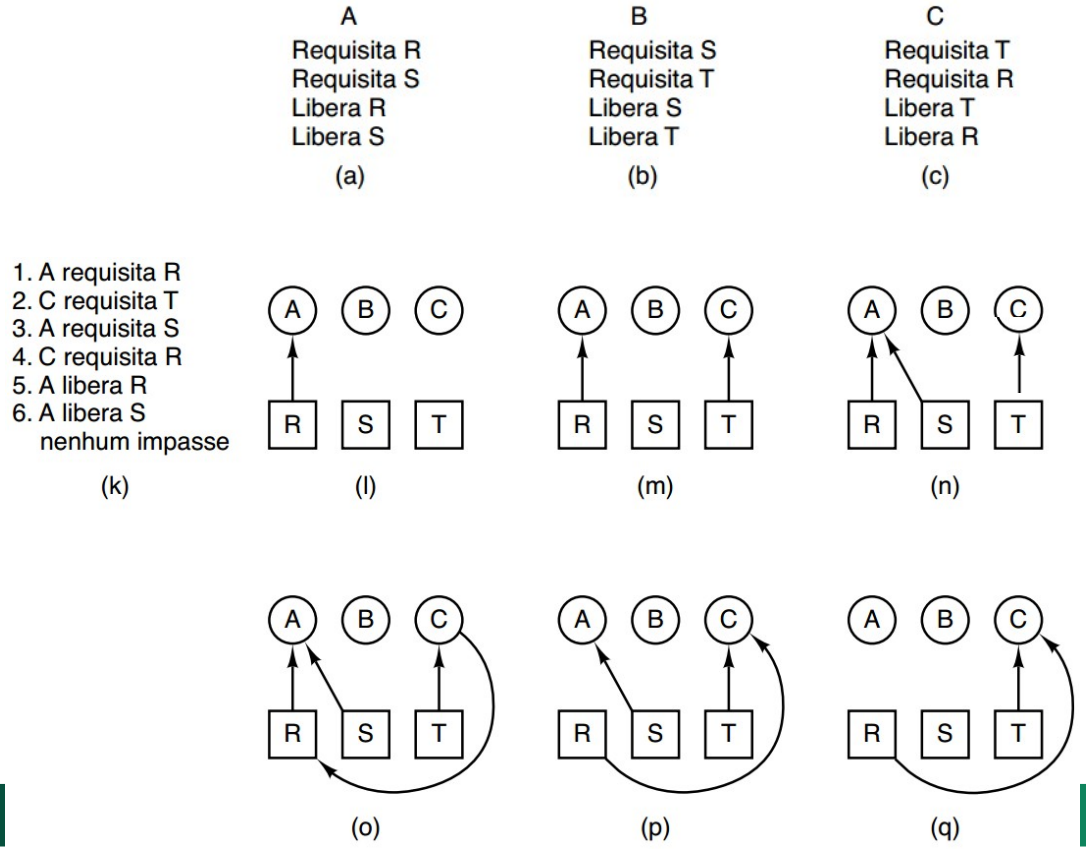
* O sistema operacional está liberado para executar qualquer processo desbloqueado a qualquer instante.

Introdução aos impasses

* Em particular, se conceder uma solicitação específica pode levar a um impasse, o sistema operacional pode simplesmente suspender o processo sem conceder a solicitação (isto é, apenas não escalonar o processo) até que seja seguro.



Introdução aos impasses



* Após o passo (q), S pode ser concedido ao processo B porque A foi concluído e C tem tudo de que ele precisa. Mesmo que B bloqueie quando solicita T, nenhum impasse pode ocorrer. B apenas esperará até que C tenha terminado.

Introdução aos impasses

Os grafos de recursos são úteis ou não?

Introdução aos impasses

Em geral, quatro estratégias são usadas para lidar com impasses:

1. Simplesmente ignorar o problema. Se você o ignorar, quem sabe ele ignore você;
2. Detecção e recuperação. Deixe-os ocorrer, detecte-os e tome as medidas cabíveis;
3. Evitar dinamicamente pela alocação cuidadosa de recursos;
4. Prevenção, ao negar estruturalmente uma das quatro condições.

Algoritmo do avestruz

- A abordagem mais simples é o algoritmo do avestruz: enfie a cabeça na areia e finja que não há um problema;

  Matemática: inaceitável e os impasses devem ser evitados a todo custo;

  Engenharia: qual a frequência que se espera que o problema ocorra? qual a frequência com que ocorrem quedas no sistema por outras razões? e quão sério um impasse é realmente?

- Se a frequência é baixa, vale a pena pagar um alto preço em termos de desempenho ou conveniência para eliminar os impasses?

Algoritmo do avestruz

- Considere um sistema operacional que bloqueia o chamador quando uma chamada de sistema open para um dispositivo físico não pode ser executada porque o dispositivo está ocupado (decisão do driver);

  Processo A: sucesso em abrir a unidade de Blu-ray e tentativa de abrir impressora;

  Processo B: sucesso em abrir a impressora e tentativa de abrir Blu-ray.

- Quais as possibilidades? Bloqueio ou código de erro.

Objetivo: questões de fixação (1)

37. Qual das alternativas a seguir melhor define uma Região Crítica em Sistemas Operacionais?

(a) Um trecho de programa que deve ser executado em paralelo com a Região Crítica de outro programa.

(b) Um trecho de programa cujas instruções podem ser executadas em paralelo e em qualquer ordem.

(c) Um trecho de programa onde existe o compartilhamento de algum recurso que não permite o acesso concomitante por mais de um programa.

(d) Um trecho de programa onde existe algum recurso cujo acesso é dado por uma prioridade.

(e) Um trecho de programa onde existe algum recurso a que somente o sistema operacional pode ter acesso.

Objetivo: questões de fixação (2)

32. [**FU**] Qual dos seguintes mecanismos é o **menos** recomendado para se implementar regiões críticas em sistemas operacionais?

(a) Semáforo
(b) Espera ocupada
(c) Troca de mensagens
(d) Monitores
(e) Variáveis de condição

Objetivo: questões de fixação (3)

Questão 52

Analise as seguintes afirmativas.

I. Condições de corrida podem ocorrer se múltiplas *threads* fazem leituras de um dado compartilhado, mesmo que nenhuma realize escritas.

II. O uso de *mutex* para a exclusão mútua em seções críticas garante que não haja condição de corrida, porém pode ocasionar *deadlocks* se não for corretamente empregado.

III. Monitores são baseados em um tipo abstrato de dados e um controle de acesso aos dados. Apenas funções do monitor acessam os dados e apenas uma *thread* ou processo pode executar funções de um monitor por vez.

IV. Semáforos têm duas operações, P( ) e V( ), sendo que apenas a operação P( ) pode bloquear um processo ou *thread*.

A análise permite concluir que

A) apenas as afirmativas I, II e III são verdadeiras.

B) apenas as afirmativas I, III e IV são verdadeiras.

C) apenas as afirmativas II e IV são verdadeiras.

D) apenas as afirmativas II, III e IV são verdadeiras.

E) nenhuma das afirmativas é verdadeira.

Objetivo: questões de fixação (4)

Em um sistema operacional multitarefa, três processos compartilham dois recursos. Cada um destes processos possui, no mínimo,

(A) seis seções críticas.

(B) quatro seções críticas.

(C) três seções críticas.

(D) duas seções críticas.

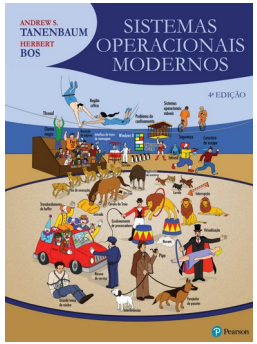
(E) uma seção crítica.

Objetivo: questões de fixação (5)

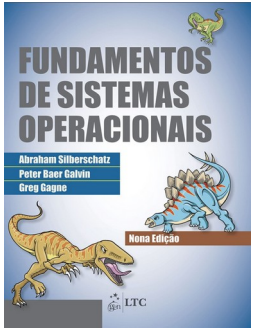
QUESTÃO 46 – Em um sistema computacional multiprocessado, onde o sistema operacional realiza escalonamento de tarefas do tipo *preemptivo*, três processos (P1, P2 e P3) compartilham recursos (R1, R2 e R3). Os processos P1 e P2 concorrem entre si ao acesso do recurso R1, enquanto P2 e P3 concorrem entre si ao acesso dos recursos R2 e R3. Os recursos R1 e R3 são *preemptíveis*, ou seja, podem sofrer preempção; R2 é um recurso *não preemptível*. Todos os três processos usam o mesmo mecanismo de exclusão mútua para garantir acesso exclusivo em suas seções críticas. Com base nesse cenário, é correto afirmar que:

A) Não é possível ocorrer *deadlock* entre os três processos.
B) É possível ocorrer *deadlock* entre P1 e P2.
C) É possível ocorrer *deadlock* entre P2 e P3.
D) É possível ocorrer *deadlock* entre P1 e P3.
E) É possível ocorrer *deadlock* com uma espera circular entre P1, P2 e P3.

Referências



TANENBAUM, Andrew S.; BOS, Herbert. Sistemas operacionais modernos. 4. edição. São Paulo: Pearson, 2016. xviii, 758 p. ISBN 9788543005676.



SILBERSCHATZ, Abraham.; GALVIN, Peter Baer.; GAGNE, Greg. Fundamentos de sistemas operacionais. 9. edição. Rio de Janeiro: LTC, 2015.

O modelo desta apresentação foi criado pelo Slidesgo.
Agradeço ao Prof. Bruno Kimura da Universidade Federal de São Paulo pelo material disponibilizado.