



# DEADLOCKS



# Deadlocks

Introdução;

Ocorrência;

Prevenção;

Deteccção;

Correção.

# Deadlocks

Introdução;

Um processo é dito em *deadlock* quando está esperando por um evento que jamais ocorrerá.

Quando vários processos competem por um número finito de recursos, um processo solicita um recurso e o recurso não está disponível, assim, o processo entra em estado de espera e jamais muda de estado, pois ali existem outros processos que esperam pelo mesmo recurso.

# Deadlocks

Introdução;

Os recursos são divididos em vários tipos:

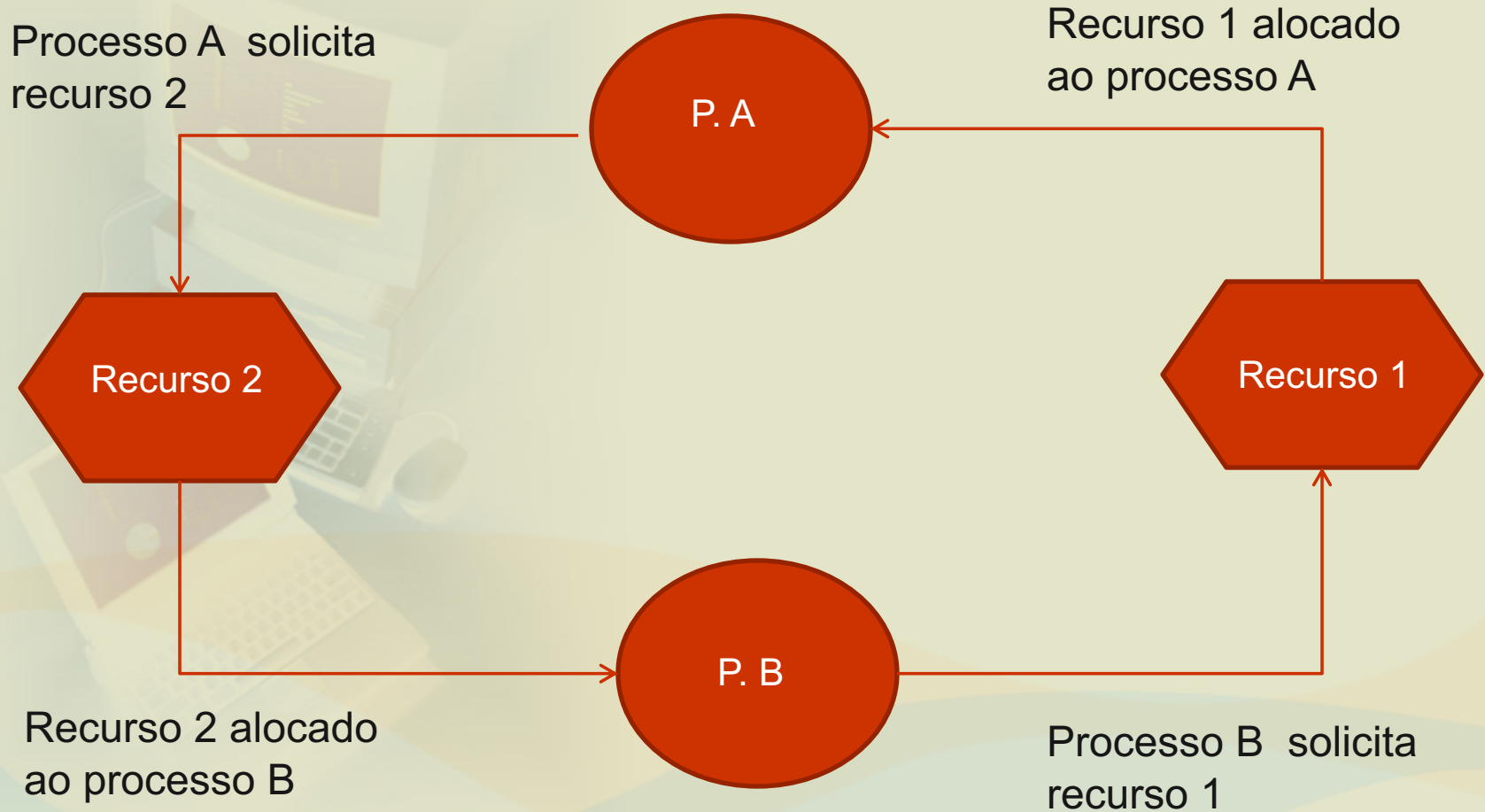
Espaço de memória;

Ciclos de CPU;

Arquivos;

Dispositivos de I/O.

# Deadlocks



# Deadlocks

Para que ocorram situações de *deadlock* em um sistema, são necessárias pelo menos quatro condições:

Exclusão mútua;

Posse e espera;

Não-preempção;

Espera circular.

# Deadlocks

Exclusão mútua;

Cada recurso só pode ser alocado a um único processo em um determinado instante;

Se outro processo solicitar esse recurso, o processo solicitante deverá ser retardado até que o recurso tenha sido liberado.

# Deadlocks

## Posse e Espera:

Um processo, além dos recursos já alocados, pode estar esperando por outros recursos;

Deve haver um processo que esteja mantendo pelo menos um recurso e esteja esperando para obter recursos adicionais que estejam sendo mantidos por outros processos no momento.



# Deadlocks

## Não-preempção:

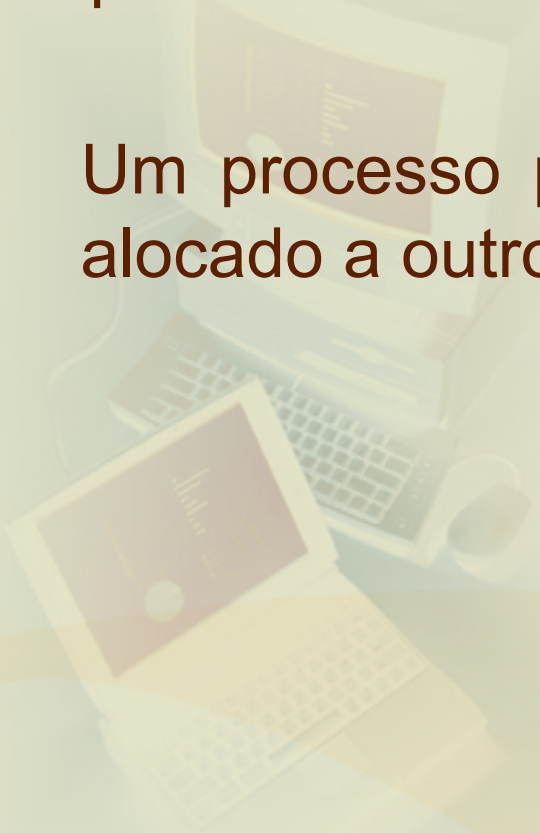
Um recurso não pode ser liberado de um processo só porque outros processos desejam o mesmo recurso;

Os recursos não podem sofrer preempção, ou seja, um recurso só pode ser liberado voluntariamente pelo processo que o mantém, depois que este processo tiver completado sua tarefa.

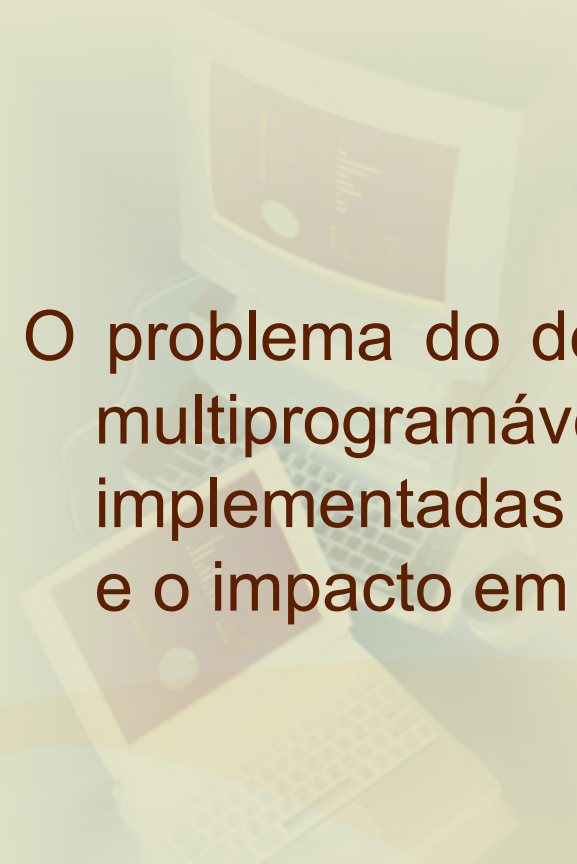
# Deadlocks

## Espera Circular:

Um processo pode ter de esperar por um recurso alocado a outro processo e vice-versa.



# Deadlocks

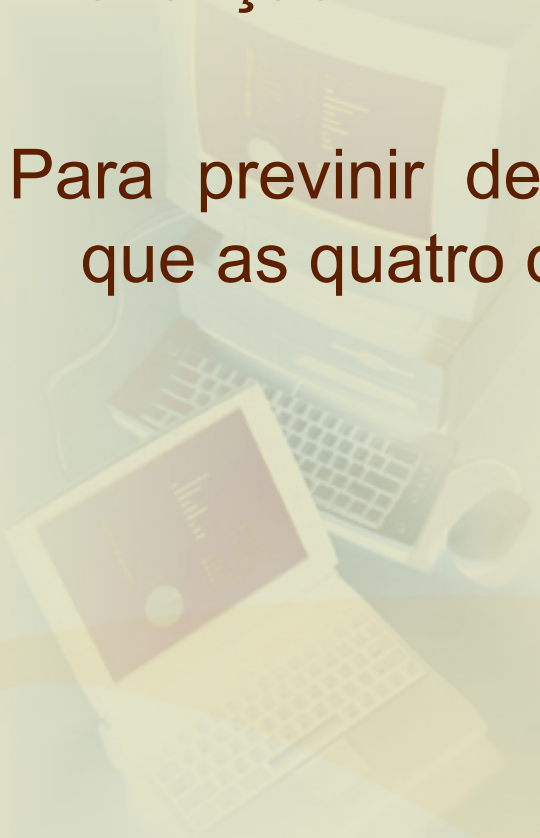


O problema do deadlock existe em qualquer sistema multiprogramável; no entanto, as soluções implementadas devem considerar o tipo de sistema e o impacto em seu desempenho.

# Deadlocks

Prevenção:

Para prevenir deadlocks, deve-se evitar ao máximo que as quatro condições citadas se satisfaçam.



# Deadlocks

## Prevenção – Exclusão Mútua

A ausência da exclusão mútua certamente acaba com o problema de deadlock, pois nenhum processo terá que esperar para ter acesso a um recurso, mesmo que já esteja em uso por um outro processo

Entretanto a falta da exclusão mútua em processos concorrentes, gera inconsistências sérias em nível de processos e sistema.

# Deadlocks

## Prevenção – Posse e espera

Se evitar que processos que já possuem recursos garantidos solicitem novos recursos, o problema de deadlock também será evitado;

Uma maneira de implementar este mecanismo é, sempre que um processo necessitar de recursos para sua execução, ele os requisite antes de começar sua execução.

# Deadlocks

## Prevenção – Posse e espera

Se todos os recursos necessários estiverem disponíveis, o processo poderá alocá-los e iniciar sua execução, caso contrário, nenhum recurso será alocado e o processo ficará em estado de espera;

Porém este mecanismo apresenta algumas formas de desperdício na utilização dos recursos do sistema, por exemplo: suponha que um processo leve 2 horas para processar os dados em uma fita magnética, e ao término imprima os resultados.

# Deadlocks

Prevenção – Posse e espera

Como o processo deve alocar os recursos antes de sua execução, a impressora (recurso) ficará alocada por 2 horas a um processo sem ser utilizada na maioria do tempo.

Outro problema é determinar a quantidade de recursos que um processo deverá alocar antes de sua execução.

Por fim, o mais grave é a ocorrência de starvation (recursos necessários não estarem disponíveis ao mesmo tempo)



# Deadlocks

## Prevenção – Não-preempção

Esta condição pode ser evitada quando permitimos que um recurso seja retirado de um processo, em caso de outro processo necessitar do mesmo recurso.

A possível ocorrência de starvation pode ser um problema deste mecanismo, pois quando o processo garante alguns recursos necessários à sua execução e o sistema os libera em seguida através de preempção.

# Deadlocks

## Prevenção – Espera circular

A última forma de se evitar o deadlock é excluir a possibilidade da espera circular.

Uma forma de implementar esse mecanismo é forçar o processo a ter apenas um recurso de cada vez, sendo que se ele precisar de um outro recurso, deve liberar primeiramente o que está utilizando.

# Deadlocks

## Detecção

Em sistemas que não possuam mecanismos que previnam a ocorrência de deadlocks, é necessário um esquema de detecção e correção do problema.

Os sistemas operacionais, para detectar deadlocks, devem manter estruturas de dados capazes de identificar cada recurso do sistema, o processo que o está alocando e os processos que estão à espera da liberação do recurso.

Toda vez que um recurso é alocado ou liberado por um processo, a estrutura deve ser atualizada.

# Deadlocks

## Detecção

Geralmente, os algoritmos que implementam esse mecanismo verificam a existência da espera circular, percorrendo toda a estrutura sempre que um processo solicita um recurso e ele não pode ser imediatamente garantido.

# Deadlocks

Correção:

Detectado o deadlock ele deverá ser tratado. Uma solução bastante comum é a eliminação do processo envolvido em deadlock e seus recursos desalocados, quebrando assim a espera circular;

Um outro tipo de correção envolve a liberação de apenas alguns recursos alocados aos processos para outros processos, até que o ciclo de espera termine. Para que isso ocorra, o sistema deve suspender o processo, liberar seus recursos e, após a solução do problema, retornar à execução do processo sem perder o processamento já realizado. Este mecanismo é conhecido como *rollback*.

# Deadlocks

Resumo:

Deadlock é um problema potencial em qualquer S.O.

Ele ocorre quando em um grupo de processos cada um recebe o direito de acesso exclusivo a alguns recursos e ainda deseja outro recurso pertencente a outro processo do grupo. Todos serão bloqueados e nenhum vai executar novamente.

Deadlocks podem ser evitados por meio do controle de quais estados são seguros e quais são inseguros:

Seguros: apresenta sequência de eventos garantindo a conclusão de todos os processos;

Inseguros: não apresenta esta garantia