UniEVANGÉLICA

UNIVERSIDADE EVANGÉLICA DE GOIÁS

SINCRONIZAÇÃO DE PROCESSOS E THREADS

Sistemas Operacionais

Engenharia de Software

Prof. Jeferson Silva

SINCRONIZAÇÃO

Processo de coordenar a execução de processos ou threads concorrentes que compartilham recursos em um ambiente de computação.

Isso é necessário para garantir que o acesso aos recursos seja feito de forma ordenada e coordenada, evitando problemas como condições de corrida, onde múltiplos processos ou threads tentam acessar um recurso compartilhado ao mesmo tempo, resultando em resultados incorretos ou indeterminados.

SINCRONIZAÇÃO

É fundamental para garantir a consistência e a corretude do sistema como um todo.

Ela envolve o uso de técnicas e mecanismos de sincronização, como semáforos, mutexes (mutual exclusion), monitores e variáveis de condição, para garantir que os processos ou threads compartilhem recursos de forma ordenada e segura.

SINCRONIZAÇÃO

Técnicas de sincronização permitem que os processos ou threads cooperem entre si, evitando conflitos e garantindo que as operações sejam executadas corretamente em relação aos recursos compartilhados.

A sincronização em sistemas operacionais também aborda questões como a prevenção, detecção e recuperação de deadlocks, que são situações em que os processos ou threads ficam bloqueados indefinidamente, aguardando recursos que nunca são liberados.

SINCRONIZAÇÃO

É uma parte fundamental do projeto de sistemas concorrentes e paralelos, e é uma habilidade essencial para os programadores e desenvolvedores que trabalham com sistemas operacionais e programação concorrente.

PROBLEMA DA SEÇÃO CRÍTICA

É um desafio enfrentado em sistemas operacionais e programação concorrente, que ocorre quando múltiplos processos ou threads tentam acessar simultaneamente uma região de código ou um recurso compartilhado, conhecido como seção crítica, levando a resultados incorretos ou indeterminados.

PROBLEMA DA SEÇÃO CRÍTICA

Em outras palavras, o problema da seção crítica ocorre quando dois ou mais processos ou threads tentam executar uma seção crítica ao mesmo tempo, resultando em uma condição de corrida.

Isso pode levar a inconsistências nos dados, comportamento imprevisível do programa ou até mesmo falhas no sistema.

PROBLEMA DA SEÇÃO CRÍTICA

Para garantir a corretude e consistência do sistema, é necessário implementar mecanismos de sincronização adequados para gerenciar o acesso à seção crítica.

Esses mecanismos devem garantir que apenas um processo ou thread por vez possa acessar a seção crítica, enquanto os demais devem esperar por sua vez.

PROBLEMA DA SEÇÃO CRÍTICA

Além disso, os mecanismos de sincronização devem evitar problemas como deadlocks, onde múltiplos processos ou threads ficam bloqueados indefinidamente, aguardando a liberação da seção crítica.

SOLUÇÃO PARA O PROBLEMA DA SEÇÃO CRÍTICA

Uma solução para o problema da seção crítica deve satisfazer aos três requisitos a seguir:

- Exclusão Mútua
- Progresso
- Espera limitada

EXCLUSÃO MÚTUA

Se o processo P_i está executando sua seção crítica, então nenhum outro processo pode executar sua seção crítica.

PROGRESSO

Se nenhum processo está executando sua seção crítica e algum processo quer entrar em sua seção crítica, então somente aqueles processos que não estão executando suas seções remanescentes podem participar da decisão de qual entrará em sua seção crítica a seguir, e essa seleção não pode ser adiada indefinidamente.

ESPERA LIMITADA

Há um limite, ou fronteira, para quantas vezes outros processos podem entrar em suas seções críticas após um processo ter feito uma solicitação para entrar em sua seção crítica e antes de essa solicitação ser atendida.

MUTEX

Mutex é uma abreviação para "mutual exclusion" (exclusão mútua) e é um mecanismo de sincronização utilizado em sistemas operacionais e programação concorrente para garantir que apenas um processo ou thread por vez possa acessar um recurso compartilhado, evitando condições de corrida e garantindo o funcionamento correto do sistema.

MUTEX

Um mutex é uma variável especial que possui dois estados possíveis: bloqueado ou desbloqueado.

Quando um processo ou thread deseja acessar o recurso compartilhado, ele deve primeiro obter o mutex associado a esse recurso.

Se o mutex estiver desbloqueado, o processo ou thread pode adquiri-lo e torná-lo bloqueado, indicando que está atualmente acessando o recurso.

MUTEX

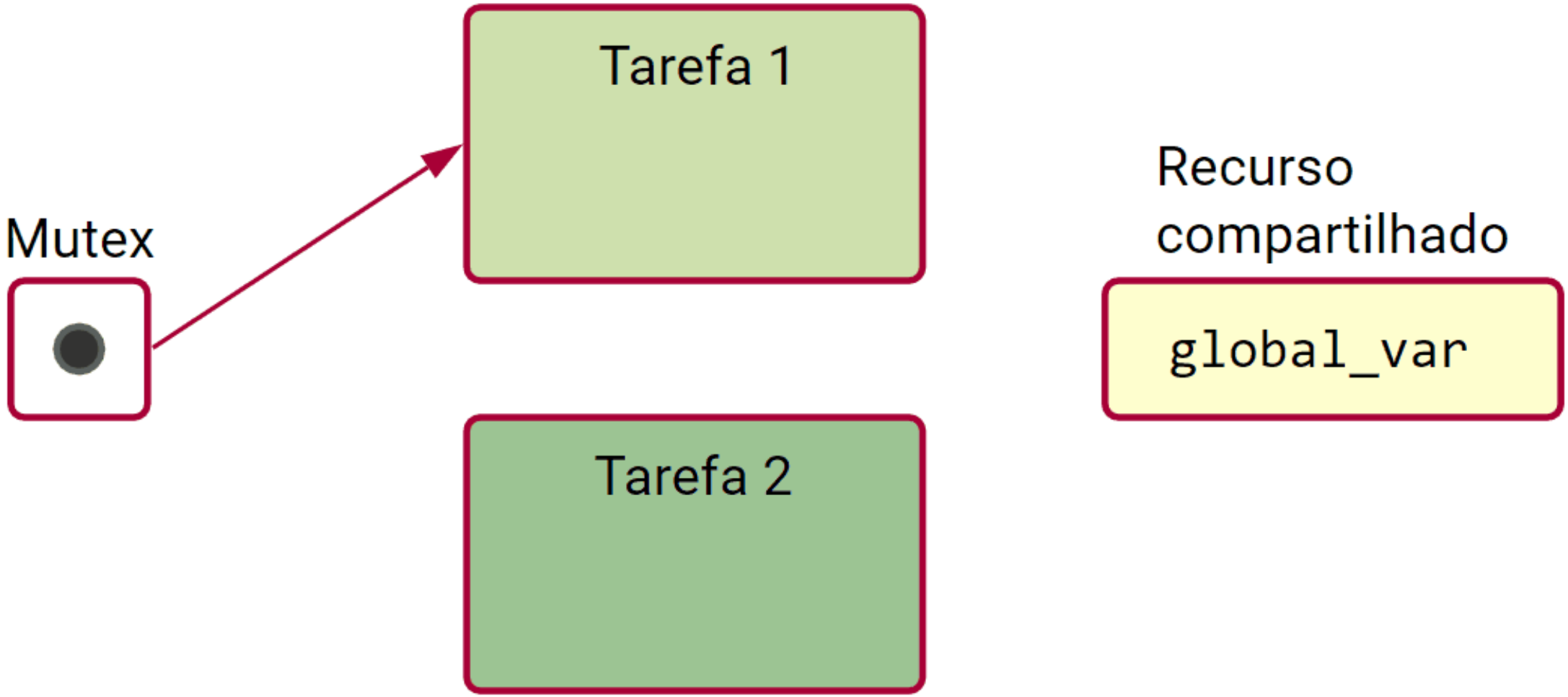
Caso contrário, se o mutex já estiver bloqueado, o processo ou thread deverá esperar até que o mutex seja desbloqueado por outro processo ou thread.

MUTEX

A exclusão mútua é garantida pelo fato de que apenas um processo ou thread pode possuir o mutex em seu estado bloqueado de cada vez.

Isso impede que outros processos ou threads possam acessar o recurso compartilhado simultaneamente, evitando condições de corrida e garantindo a consistência dos dados.

MUTEX



SEMÁFOROS

É um tipo de variável especial que é usada como um mecanismo de sincronização para coordenar o acesso a recursos compartilhados entre processos ou threads concorrentes.

Ele foi introduzido pelo cientista da computação Edsger Dijkstra e é amplamente utilizado em programação concorrente para garantir a exclusão mútua e a coordenação entre processos ou threads.

SEMÁFOROS

Um semáforo pode ter um valor inteiro não negativo e pode ser manipulado através de duas operações básicas:

- Wait (down ou P)
- Signal (up ou V)

SEMÁFORO – WAIT

Se o valor do semáforo for maior que zero, ele é decrementado e o processo ou thread continua sua execução.

Caso contrário, se o valor do semáforo for igual a zero, o processo ou thread é bloqueado até que o valor do semáforo se torne estritamente maior que zero, indicando que o recurso compartilhado está disponível.

SEMÁFORO – SIGNAL

O valor do semáforo é incrementado, indicando que o recurso compartilhado foi liberado.

Se houver processos ou threads bloqueados em uma operação de Wait, um deles é desbloqueado e pode prosseguir com sua execução.

EXEMPLO DE SINCRONIZAÇÃO