



Unidade 2

Seção 2

Sistemas Operacionais



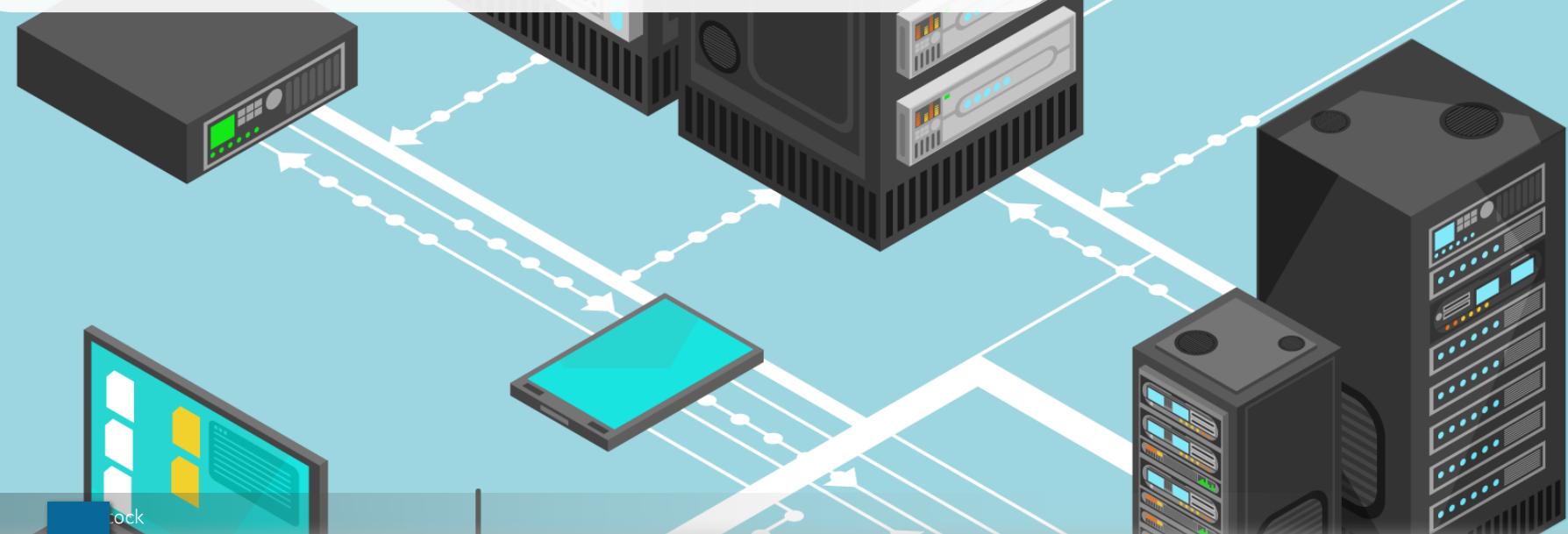


Weaula 2

Comunicação entre processos



Nessa webaula veremos algumas questões sobre a comunicação entre processos, tais como de disputa, regiões críticas e exclusão mútua com espera ociosa. Além disso, estudaremos os mecanismos de sincronização que resolvem a exclusão mútua: dormir e acordar, semáforos, monitores e troca de mensagens.





Condições de Disputa ou Condições de Corrida

Condições de disputa ou condições de corrida ocorrem quando dois ou mais processos estão lendo ou escrevendo algum dado compartilhado e o resultado final depende das informações de quem e quando executa, podendo gerar dados inconsistentes.





Regiões críticas

Para evitar as condições de disputa, é necessário definir maneiras que impeçam que mais de um processo leia e escreva ao mesmo tempo na memória compartilhada.

Segundo Tanenbaum, (2007) para termos uma boa solução, é necessário satisfazer quatro soluções:

Nunca dois ou mais processos podem estar simultaneamente em suas regiões críticas;

Nada pode ser afirmado sobre o número e a velocidade de CPUs;

Nenhum processo executando em uma região crítica pode bloquear outros processos;

Nenhum processo pode esperar eternamente para entrar em sua região crítica.





Esses métodos são chamados de exclusão mútua. Para realizá-los, podemos lançar mão das seguintes estratégias:

Exclusão mútua com
espera ociosa

Dormir e Acordar

Semáforos

Monitores

Troca de Mensagens





Exclusão Mútua com Espera Ociosa

Existem alguns métodos que determinam que quando um processo está em sua região crítica, nenhum outro pode invadi-la. Explore a galeria.

Desabilitando Interrupções

Nesta solução, cada processo desabilita todas as interrupções assim que entra em sua região crítica e as reabilita antes de sair dela. Desta forma, a CPU não será alternada para outro processo.





Dormir e Acordar

Para resolver uma espera ociosa, são realizadas chamadas aos sistemas *sleep* (Dormir) e *wakeup* (Acordar), que bloqueiam/desbloqueiam o processo em vez de gastar tempo de CPU com a espera ociosa.

A chamada *sleep* faz com que o processo que a chamou durma até que outro processo o desperte e a chamada *wakeup* acorda um processo.





Semáforos

Um semáforo é uma variável inteira que realiza duas operações. Clique nas abas a seguir:

DOWN

Decrementa uma unidade do valor do semáforo

UP

Incrementa uma unidade ao valor do semáforo





As instruções DOWN e UP são indivisíveis e executadas no processador. Os semáforos podem ser classificados como:

Binários, também conhecidos como mutexes, podem receber os valores 0 ou 1.

Contadores podem receber qualquer valor inteiro positivo, além do 0.





Monitores

Um monitor é uma coleção de procedimentos, variáveis e estrutura de dados agrupados em um módulo ou pacote.

Quando um processo chama um procedimento de um monitor, é verificado se outro processo está ativo. Caso esteja, o processo que o chamou é suspenso até que o outro deixe o monitor, caso contrário, o processo que chamou poderá entrar.

Com o uso de monitores, a exclusão mútua é garantida, pois somente um processo pode estar ativo dentro do monitor em um mesmo instante e os demais processos ficam bloqueados até que possam estar ativos no monitor..





Troca de Mensagens

Esse método utiliza duas chamadas ao sistema. Clique nas abas.

Send

Envia uma mensagem para um determinado destino.

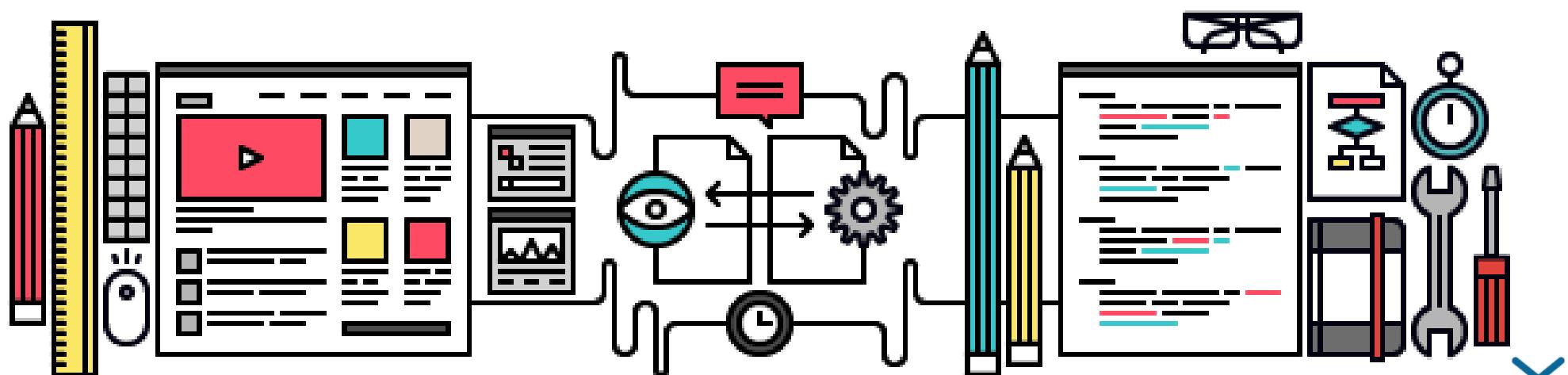
Receive

Recebe uma mensagem de uma determinada origem.





Vimos nesta seção que para resolver o problema de exclusão mútua existem as seguintes soluções: Exclusão Mútua com Espera Ociosa, Dormir e Acordar, Semáforos, Monitores, Troca de Mensagens. Esses métodos são equivalentes quando implementados em um único processador.







Bons estudos!

