Solução de Exclusão Mútua com Espera Ocupada

A **exclusão mútua** é uma técnica usada para evitar que vários threads ou processos acessem simultaneamente uma seção crítica, garantindo consistência dos dados. Algumas soluções utilizam **espera ocupada** , o que significa que um processo fica em um loop contínuo verificando se pode acessar o recurso.

1. Soluções com Espera Ocupada

A) Variável de Bloqueio

- **Como funciona:** Um processo define uma variável de bloqueio como `true`antes de entrar na seção crítica e define como `false`ao sair.
- **Pontos Fracos:** Podem causar **condição de disputa** (race Condition), onde dois processos verificam a variável ao mesmo tempo e ambos entram na seção crítica.

B) Algoritmo de Dekker

- **Como funciona:** Usa uma variável de turno e sinalizadores individuais para cada processo indicam sua intenção de acessar a seção crítica.
- **Pontos Fortes:** Garantir exclusão mútua e evitar impasse.
- **Pontos Fracos:** Dificuldade em escalabilidade para mais de dois processos.

C) Algoritmo de Peterson

- **Como funciona:** Usa uma variável de turno e sinalizadores para controle, garantindo que apenas um processo entre na seção crítica de cada vez.
- **Pontos Fortes:** Simples e eficaz para dois processos.
- **Pontos Fracos:** Não escala bem para processos múltiplos.

D) Teste e ajuste (TS)

- **Como funciona:** Use uma instrução atômica da CPU para definir e verificar uma variável de bloqueio.
- **Pontos Fortes:** Implementação eficiente no nível do hardware.
- **Pontos Fracos:** Pode levar uma **espera ocupada** , gastando tempo de CPU.

E) Comparação e troca (CAS)

- **Como funciona:** Permite alterar um valor na memória apenas se ele não foi modificado por outro processo.
- **Pontos Fortes:** Rápido e amplamente utilizado em sistemas modernos.
- **Pontos Fracos:** Se houver muita concorrência, pode haver muitas tentativas fracassadas, aumentando o tempo de execução.

---

2. Problemas da Espera Ocupada

- **Ineficiente** : Processos ficam verificando constantemente a permissão, desperdiçando CPU.
- **Pode causar fome** : Um processo pode ficar indefinidamente esperando a liberação do recurso.
- **Afeta escalabilidade** : Com muitos processos, a contenção aumenta.

---

3. Alternativas Melhores

- **Semáforos** : Bloqueiam o processo em espera, sem desperdício de CPU.
- **Monitores** : Abstração de alto nível que encapsula a sincronização dentro dos objetos.
- **Fechaduras sem espera ocupada** : Como mutexes, que suspendem um thread em vez de consumir CPU.