Nomes: Roberto Zhou e Matheus Bortoli Da Silva

1. Filósofos

O protocolo evita o impasse porque elimina a condição de espera circular.
Cada filósofo sempre pega primeiro o garfo de menor índice e depois o de maior índice, criando uma ordem global fixa para todos.
Com isso, não existe um ciclo de espera entre os filósofos, já que ninguém pode ficar aguardando indefinidamente por um garfo que outro está segurando.
Dessa forma, o sistema continua funcionando normalmente, sem travar, e todos os filósofos conseguem pensar e comer em momentos diferentes, evitando o deadlock.

Reprodução do deadlock e análise segundo Coffman

No experimento proposto, utilizamos duas threads e dois recursos (locks A e B).
A Thread 1 tenta adquirir primeiro o lock A e depois o lock B.
A Thread 2 faz o inverso: primeiro tenta pegar o lock B e depois o lock A.

```
Filósofo 2 está com fome.
Filósofo 1 está com fome.
Filósofo 0 está com fome.
Filósofo 3 está com fome.
Filósofo 2 pegou o garfo esquerdo.
Filósofo 3 pegou o garfo esquerdo.
Filósofo 0 pegou o garfo esquerdo.
Filósofo 4 está com fome.
Filósofo 1 pegou o garfo esquerdo.
Filósofo 4 pegou o garfo esquerdo.
```

Depois disso, não há mais progresso: nenhuma das threads libera o lock que possui porque estão esperando uma pela outra. O programa continua executando, mas não imprime mais nada, indicando que ocorreu um deadlock real.

Relação com as condições de Coffman

O deadlock produzido satisfaz as quatro condições necessárias definidas por Coffman:
Exclusão mútua: cada lock só pode ser utilizado por uma thread por vez.
Hold and Wait (manter-e-esperar): cada thread segura um lock enquanto espera outro.
Não preempção: os locks não podem ser retirados à força; a thread só libera voluntariamente.
Espera circular: a Thread 1 espera B, que está com a Thread 2, enquanto a Thread 2 espera A, que está com a Thread 1, formando um ciclo.
Como todas as condições estão presentes, o deadlock ocorre.

2. Implementação corrigida e explicação da solução

Para resolver o problema, aplicamos a técnica de hierarquia de recursos, que também é uma solução clássica no problema do Jantar dos Filósofos.

A ideia é simples: todas as threads devem adquirir os recursos sempre na mesma ordem. No caso do trabalho, definimos que todos devem pegar primeiro o lock A e só depois o lock B.
Mesmo que duas threads tentam acessar os recursos ao mesmo tempo, nenhuma delas pode ficar presa esperando em um ciclo, porque a ordem de aquisição passa a ser sempre previsível.

Condição que foi eliminada:
Ao impor essa ordem fixa, eliminamos a condição de espera circular, que é uma das quatro condições de Coffman necessárias para o deadlock ocorrer.
Sem a espera circular, mesmo que as outras três condições continuem existindo (exclusão mútua, manter-e-esperar e não preempção), o deadlock não acontece mais.

```
Esperado=2000000, Obtido=324243, Tempo=0,044s
```

3. Relação com o Jantar dos Filósofos

O problema do deadlock encontrado nesse exercício é muito parecido com o clássico Jantar dos Filósofos.
No jantar, cada filósofo precisa de dois garfos para comer. Se todos pegarem um garfo e ficarem esperando o outro, o sistema trava e ninguém progride exatamente como aconteceu com nossas duas threads e seus dois locks.
No Jantar dos Filósofos, uma solução tradicional é definir uma ordem fixa para pegar os garfos. Nenhum filósofo pega o garfo da direita antes do da esquerda se a ordem não permitir, Isso impede que se forme um ciclo de dependências.
No nosso caso fizemos a mesma coisa: definimos que todos os recursos devem ser adquiridos sempre na mesma ordem.
Essa abordagem impede que uma thread fique esperando eternamente por um recurso que está sendo segurado pela outra, já que ambas passam a seguir a mesma hierarquia.
Com isso evita impasse e inanição
Evita inanição (fome) porque todas as threads seguem o mesmo protocolo e não existe uma prioridade fixa que sempre favoreça uma delas.
A estratégia garante que, quando uma thread liberar os recursos, outra poderá avançar sem risco de travar o sistema.
Assim como na Parte 1 (filósofos) usamos um protocolo para evitar bloqueios, e na Parte 2 usamos semáforos para evitar corrida, aqui aplicamos de novo os mesmos princípios: organizar o acesso aos recursos, evitar ciclos e garantir progresso.

```
[T1] tentando adquirir A...
T2: tentando adquirir B...
[T1] adquiriu A
T2: adquiriu B
[T1] tentando adquirir B...
T2: tentando adquirir A...
```