

## TDE-3—Performance

### DESCRIÇÃO DO PROBLEMA

O Jantar dos Filósofos é um problema clássico de concorrência usado para ilustrar situações onde múltiplos processos competem por recursos compartilhados. Os filósofos estão sentados em uma mesa circular com um garfo entre cada par. Para comer, cada filósofo precisa pegar dois garfos: o da esquerda e o da direita.

O jantar dos filósofos modela threads concorrendo por recursos. - Filósofos → threads - Garfos → locks - Comer → exige dois recursos simultâneos

Cada filósofo alterna entre.

- Pensar
- Ficar com fome
- Comer (se conseguir ambos os garfos)

Sem um controle adequado, todos podem pegar um garfo e esperar eternamente pelo outro, causando um **deadlock** (impasse).

### o que é um Deadlock (impasse)

Deadlock ocorre quando processos ficam presos esperando recursos que nunca serão liberados. Um deadlock só pode existir quando todas as quatro condições de Coffman estão presentes:

- Exclusão mútua: um recurso só pode ser usado por um processo por vez
- Manter e esperar: um processo segura recursos enquanto espera outro
- Não-preempção: recursos não podem ser retirados à força
- Espera circular: existe um ciclo fechado de dependências

Ciclo de deadlock: **F0 espera F1**

**F1 espera F2**

**F2 espera F3**

**F3 espera F4**

**F4 espera F0**

No jantar dos filósofos, é justamente a espera circular que causa o impasse.

### Fome (Starvation)

Fome ocorre quando um processo nunca progride, mesmo que o sistema continue ativo. No contexto deste problema, um filósofo pode ficar muito tempo sem comer se for repetidamente preterido enquanto tenta pegar os garfos.

Solução do Impasse:

A solução adotada segue a estratégia de ordem global de recursos: Cada filósofo sempre pega primeiro o garfo de menor índice e depois o de maior índice.

## Funcionalidade

Numeramos os garfos de 0 a 4;  
Cada filósofo identifica seus dois garfos;  
Ele transforma isso em:

```
left = min(garfo_esquerda, garfo_direita)
right = max(garfo_esquerda, garfo_direita);
```

Em seguida, sempre adquire primeiro o menor índice

## Pseudocódigo do protocolo:

```
para cada filósofo p:
  left = menor índice entre os dois garfos de p
  right = maior índice entre os dois garfos de p
  — repetir para sempre: pensar() estado[p] <- “com fome”

      adquirir(left)
      adquirir(right)

      estado[p] <- "comendo"
      comer()

      liberar(right)
      liberar(left)

      estado[p] <- "pensando"
```

---

## Prova de Ausência de Deadlock-

A solução remove a condição de espera circular, que é essencial para que um deadlock ocorra.

Por que não existe espera circular? Todos os filósofos tentam adquirir primeiro o garfo de menor índice. Portanto, um filósofo nunca fica esperando por um garfo de índice menor do que o que já possui. Isso impede completamente a formação de ciclos de dependência. **Sem espera circular → não há deadlock**

## Sobre a Fome (Starvation)

O protocolo evita deadlock, mas não garante completamente ausência de fome. No entanto, em implementações reais, esse problema pode ser mitigado com:

Semáforos justos: Semaphore(1, true);  
Filas FIFO para acesso aos garfos;  
Políticas de agendamento que garantem fairness.

Assim, todos eventualmente comem.

## Conclusão do problema

Abordamos os conceitos de deadlock e fome; Propusemos um protocolo que elimina o impasse; Justificamos como o protocolo funciona;