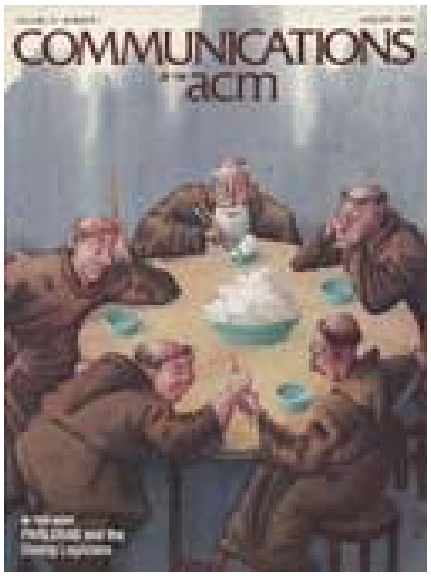
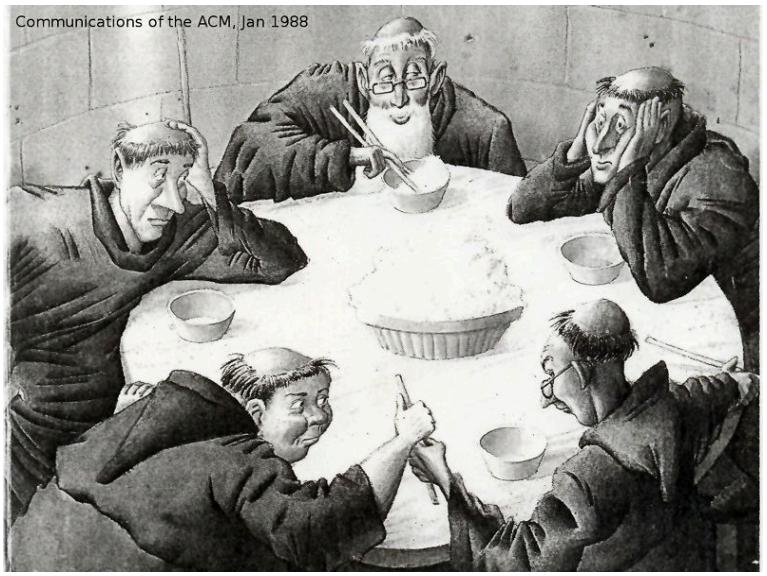
O Jantar dos filósofos

O Jantar dos Filósofos é problema clássico de sincronização proposto por Dijkstra em 1965. (https://www.cs.utexas.edu/users/EWD/ewd03xx/EWD310.PDF) . A capa da revista Communications of the ACM de Janeiro de 1988 ilustra a aplicação prática deste problema em uma situação do mundo real…





O problema:
Clássico em ciência da computação, ele ilustra os desafios de sincronização e potenciais impasses em sistemas que gerenciam recursos compartilhados. No cenário, temos vários filósofos sentados ao redor de uma mesa circular, cada um com um palito entre si e o vizinho. Cada filósofo alterna entre meditar e comer. Para comer, um filósofo precisa pegar os dois palitos adjacentes, um à sua esquerda e outro à sua direita.

O impasse ocorre quando todos os filósofos pegam o palito à sua esquerda simultaneamente e, em seguida, tentam pegar o palito à sua direita, que já está sendo segurado por outro filósofo. Como resultado, todos os filósofos ficam esperando indefinidamente pelo palito que falta, criando um impasse onde ninguém consegue comer.

Algumas possiveis soluçõe:
1. Solução do Saleiro (Resource Hierarchy Solution)
2. Solução de Inversão aleatória
3. Solução de Chandy/Misra (Politeness Algorithm)
4. Tentativa: Após pegar o primeiro hashi, filósofo “tenta” pegar o segundo. Se não conseguir, devolve o primeiro.

5. Solução do Garçom/Exemplo 4 Saleiro
6. Solução Baseada em Filas
7. Numerar os hashis: pegar o menor hashi primeiro.
8. Solução Baseada em Prioridades
9. ~~Solução com Monitor de Condições, não disponivel em C (Requer um ambiente de programação que suporte monitores, como Java.~~

Tarefa

Objetivo:
Implementar e analisar diferentes soluções para o problema do jantar dos filósofos, enfocando tanto o desempenho técnico quanto conceitos fundamentais de concorrência e sincronização

Requisitos de Implementação:

- Seleção de Soluções: Cada equipe (com no máximo 4 membros) deve abordar pelo menos uma solução por membro e implementar em C.
- Código Base: utilizar o código disponível no SIGAA.

Redigir relatório com pelo menos os seguintes componentes:

1. Explicação das Implementações:
    - Detalhar o funcionamento de cada solução implementada.
    - Incluir pseudocódigo ou trechos de código real para ilustrar pontos importantes.
2. Análise Comparativa:
    - Vantagens: Descrever os benefícios de cada solução, como eficiência, simplicidade de implementação ou robustez contra deadlocks.
    - Desvantagens: Discutir as limitações, como potencial para starvation, complexidade ou uso ineficiente de recursos.
3. Avaliação de Desempenho:
    - Desenvolver uma simulação para medir o throughput (quantos filósofos conseguem comer por unidade de tempo) e fairness (garantindo que todos os filósofos comam com frequência razoavelmente uniforme). (desabilitar as pausas para efetuar essa medição)
    - Descrever a metodologia de teste, incluindo como as pausas foram desabilitadas e como os dados foram coletados.
    - Rodar 10 vezes cada algoritmo para obter os dados, cada vez por 1 minuto.
4. Criação de Gráfico Comparativo:
5. Análise Crítica:

- Avaliar criticamente as soluções com base nos dados coletados e na teoria estudada.
- A soluções implementadas resolveram o problema de impasses? Qual ou quais das condições que são necessárias para o impasse foi ou foram removidas?