

ASYNC PROGRAMMING

Ideias centrais

- Sequencial
- Concorrente
- Paralelo

Processos e threads

- i) OS agenda as threads
- ii) Interleaving
- iii) Preemptive multitasking
- iv) Context switching

Async Programming

O quê?	Quem controla?	Como é o multitasking?
threads	OS	preemptive
async	programa	cooperativo

- Task: async concorrente

Async Programming

CONSEQUÊNCIAS DE COOPERATIVO:

- i) Garantia de execução entre pontos
- ii) Tasks “pausadas” durante execução de outra task
- iii) Implementação é mais fácil (pelo programador)

Filósofos jantando



- Proposto por Dijkstra em 1965
- **Enunciado:**

A essência do problema envolve:

- **Cinco filósofos** sentados em uma mesa, cada um com seu próprio prato.
- Um garfo entre cada prato, totalizando cinco garfos.
- Os filósofos podem alternar entre **pensar e comer**.
- Para comer o espaguete, um filósofo precisa de **dois garfos: um à sua esquerda e um à sua direita**.
- Um garfo só está disponível se seus dois vizinhos mais próximos estiverem pensando, não comendo.
- Após comer, o filósofo põe de volta ambos os garfos.

O desafio é "como projetar um regime (um algoritmo concorrente) tal que nenhum filósofo passe fome; *i.e.*, cada um possa continuar para sempre a alternar entre comer e pensar, assumindo que nenhum filósofo pode saber quando os outros podem querer comer ou pensar (uma questão de informação incompleta)

Filósofos jantando



– Situações chave:

- **Exclusão mútua:** Nenhum garfo pode ser usado simultaneamente por múltiplos filósofos.
- **Retenção de recurso (*resource holding*):** Os filósofos seguram um garfo enquanto esperam pelo segundo.
- **Não-preempção (*non-preemption*):** Nenhum filósofo pode pegar um garfo de outro.
- **Espera circular (*circular wait*):** Cada filósofo pode estar esperando pelo filósofo à sua esquerda.

- Solução: negar ao menos uma das situações acima

Filósofos jantando



– Situações chave:

- **Exclusão mútua:** Nenhum garfo pode ser usado simultaneamente por múltiplos filósofos.
- **Retenção de recurso (*resource holding*):** Os filósofos seguram um garfo enquanto esperam pelo segundo.
- **Não-preempção (*non-preemption*):** Nenhum filósofo pode pegar um garfo de outro.
- **Espera circular (*circular wait*):** Cada filósofo pode estar esperando pelo filósofo à sua esquerda.

- Solução: negar ao menos uma das situações acima

Solução de Dijkstra

Características:

- Nega a **retenção de recurso**: os filósofos pegam *ambos* os garfos ou esperam
- Cada filósofo possui:
 - Um **mutex** global.
 - Um **semáforo** individual.
 - Uma **variável de estado** (THINKING, HUNGRY, EATING).

Mecanismo:

- A função test() é usada em take_forks() e put_forks() para coordenar o acesso aos garfos.
- Garante ausência de **deadlock**.

Solução de Hierarquia

Características:

- Nega a **espera circular**: os garfos são numerados, e sempre pegos na ordem crescente.

Exemplo:

- Filósofos pegam primeiro o garfo com número **menor**, depois o **maior**.

Vantagens:

- Previne deadlocks.

Desvantagens:

- Pode exigir liberação e readquirição de recursos.
- Não garante **justiça**: um filósofo lento pode nunca comer.

Solução do Garçom

Características:

- Introduz um **garçom (ou árbitro)** que controla o acesso aos garfos.
- Um filósofo só pega garfos com a permissão do garçom.

Mecanismo:

- O garçom permite a um filósofo pegar **ambos os garfos ou nenhum**.

Vantagens:

- Simples de entender e implementar.

Desvantagens:

- Pode reduzir o **paralelismo**: outros filósofos esperam mesmo com garfos livres.
- Nunca comer.

Solução de limitar filósofos

Proposta:

- Apenas **$n-1$ filósofos** podem sentar-se à mesa ao mesmo tempo.

Objetivo:

- Garante que pelo menos um filósofo **sempre consiga comer**.

Vantagens:

- Nega a **espera circular**.
- Evita deadlock de forma simples.

Implementação:

- Pode usar um **semáforo** com valor $n-1$ para controlar entrada na mesa.

Solução de limitar filósofos

Proposta:

- Apenas **$n-1$ filósofos** podem sentar-se à mesa ao mesmo tempo.

Objetivo:

- Garante que pelo menos um filósofo **sempre consiga comer**.

Vantagens:

- Nega a **espera circular**.
- Evita deadlock de forma simples.

Implementação:

- Pode usar um **semáforo** com valor $n-1$ para controlar entrada na mesa.

Learn more:



 vinteum

Obrigado.