Mecanismos de Sincronização com Variáveis Compartilhadas

Fundamentos dos Sistemas de Tempo Real

Rômulo Silva de Oliveira

Edição do Autor, 2018

www.romulosilvadeoliveira.eng.br/livrotemporeal

Outubro/2018

Introdução

- Problema da seção crítica é o mais frequente, mas não é o único

- Existem outras situações onde é necessário sincronizar tarefas que colaboram através de variáveis compartilhadas
  - Cenários mais complexos que a simples seção crítica

- Mecanismos adicionais além do mutex são necessários
- Por exemplo
  - Semáforos
  - Monitores

Problemas Clássicos de Sincronização

- Tarefas podem apresentar outros padrões de interação mais complexos do que a simples necessidade de exclusão mútua
- Na prática existem alguns padrões de interação entre tarefas que são encontrados com relativa frequência
  - E não podem ser resolvidos apenas com mutex

- São padrões de interação os quais podemos considerar como **padrões de projeto (*design patterns*)** da programação concorrente com variáveis compartilhadas

Problemas Clássicos de Sincronização

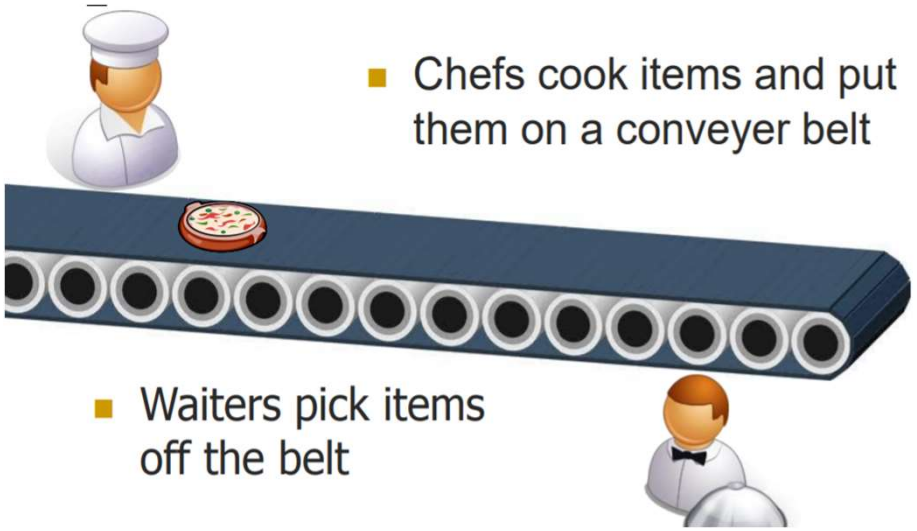
- Em geral, tais padrões incluem
  - A sempre presente necessidade de exclusão mútua
  - Relações de precedência entre tarefas

- Uma tarefa precisa esperar até que outra tarefa conclua alguma operação, para então prosseguir
- Esta espera é uma relação de sincronização entre essas tarefas
  - Requer algum mecanismo de sincronização

- Na literatura de programação concorrente os padrões de interação mais frequentes são descritos na forma de cenários exemplo que capturam a essência da situação
- Tais cenários são chamados de **problemas clássicos de sincronização** (***classic problems of synchronization***)

Problema dos produtores e consumidores (*bounded–buffer problem*)

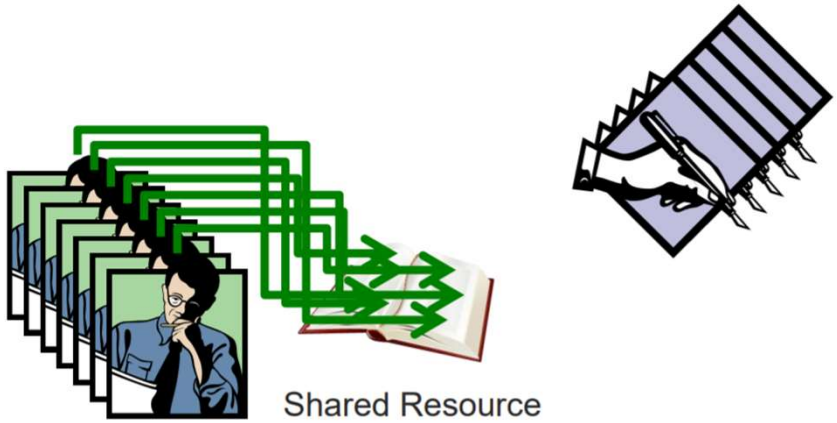
- Existe um conjunto de tarefas repetidamente produzindo dados
- e um conjunto de tarefas repetidamente consumindo estes dados
- Dados podem ser de qualquer tipo
  - Como um inteiro, um string ou uma struct
- Qualquer tarefa consumidora pode receber o dado gerado por qualquer tarefa produtora, desde que esteja livre
- A solução consiste em implementar um buffer (área de armazenamento temporário) onde produtores depositam dados e consumidores retiram dados
- O buffer consiste de um array, usado de maneira circular
  - Dados são depositados nas posições livres
  - Dados são retirados das posições ocupadas
- Necessário bloquear produtores quando o array fica lotado
- E bloquear consumidores quando o array fica vazio

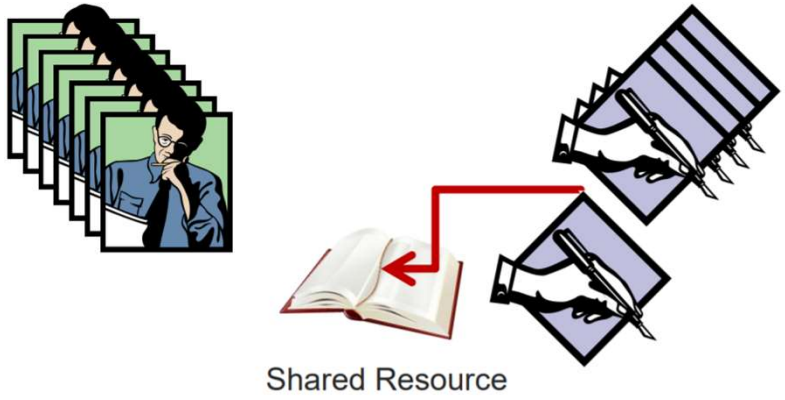


Fonte: https://courses.engr.illinois.edu/cs241/fa2012/lectures/22-ClassicSynch.pdf

Problema dos leitores e escritores (*readers–writers problem*)

- Existe um conjunto de dados que é compartilhado entre várias tarefas
- A grande maioria das tarefas deseja apenas ler estes dados
  - O que pode ser feito concorrentemente
  - Sem a necessidade de exclusão mútua entre elas
- Existem algumas tarefas que atualizam estes dados
  - Neste caso, é necessária a exclusão mútua
  - Tarefas escritoras precisam de acesso exclusivo durante a alteração dos dados
- O problema pode ser enunciado requerendo
  - Prioridade para os leitores
  - Prioridade para os escritores
  - A ordem de chegada das tarefas é respeitada



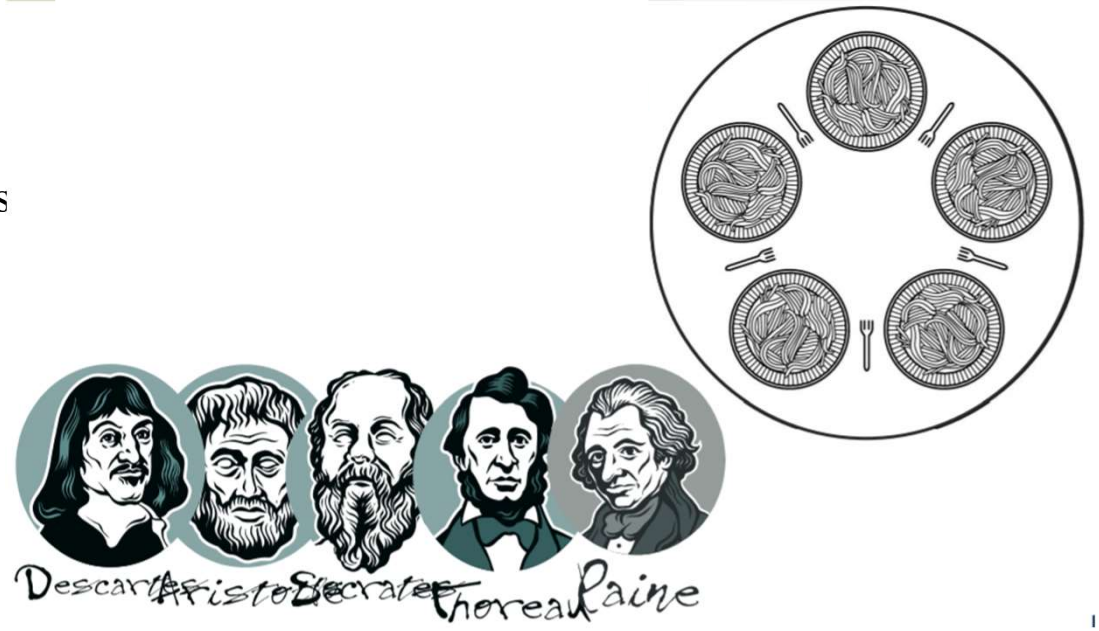


Fonte: https://courses.engr.illinois.edu/cs241/fa2012/lectures/22-ClassicSynch.pdf

Problema dos filósofos jantadores (*dining-philosophers problem*)

- Cinco filósofos (tarefas) estão sentados em torno de uma mesa circular
- Cada um deles tem um prato para seu uso exclusivo (recurso privativo)
- Existe um garfo entre cada dois filósofos (recurso compartilhado)
- Existe uma panela com macarrão no centro da mesa
- Para comer, cada filósofo usa seu prato privativo
  - Porém precisa simultaneamente dos dois garfos seu lado
  - Os quais são compartilhados com os filósofos vizinhos

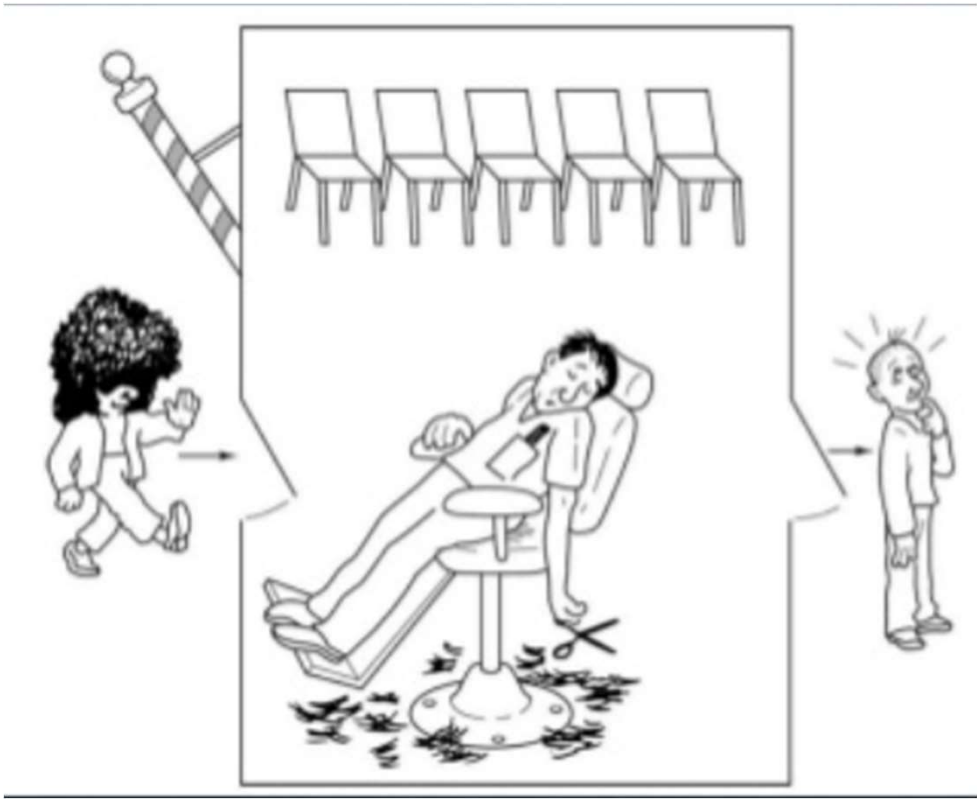
- O problema consiste em usar mecanismos de sincronização entre tarefas
  - Para impor o uso correto dos garfos
  - Sem que nenhum filósofo morra de fome (postergação indefinida)
  - Sem que o sistema entre em um impasse global (deadlock)



Fonte: https://courses.engr.illinois.edu/cs241/fa2012/lectures/22-ClassicSynch.pdf

Problema do barbeiro dorminhoco (*sleeping barber problem*)

- Existe uma barbearia onde trabalha apenas um barbeiro
- Existe uma cadeira de barbeiro usada no momento do corte da barba
- Existem algumas cadeiras de espera
  - Para os clientes usarem enquanto esperam sua vez de serem atendidos
- Quando não existem clientes, o barbeiro senta na cadeira de barbeiro e dorme
- Quando um cliente chega
  - Ele precisa acordar o barbeiro dorminhoco
  - Sentar na cadeira do barbeiro para ser atendido
- Se um cliente chega, porém o barbeiro está ocupado
  - O cliente senta em uma das cadeiras de espera
  - No caso de todas as cadeiras de espera estarem ocupadas, o cliente vai embora
- Uma solução deve respeitar as restrições do enunciado
  - Impedir postergação indefinida
  - Evitar um impasse que trave o funcionamento da barbearia

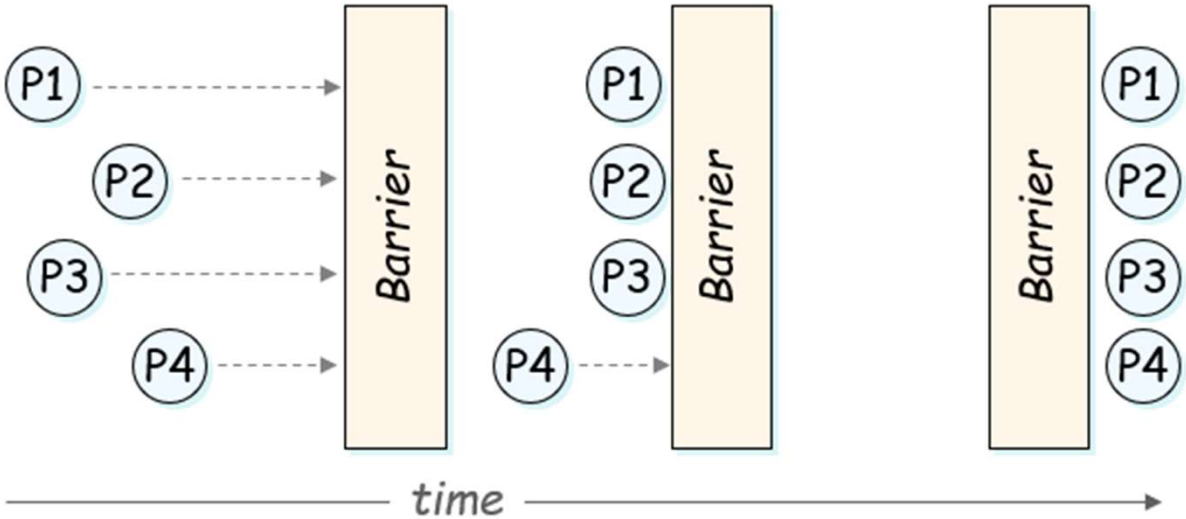


Fonte: https://pt.slideshare.net/DhananjaysinhJhala/sleeping-barber-problem

Problema da barreira (*barrier problem*)

- Existem várias tarefas que cooperam dividindo o trabalho a ser feito
- Existe um ponto do algoritmo (a barreira) no qual todas as tarefas precisam chegar antes que qualquer uma delas possa prosseguir
- A barreira é um ponto de encontro de todas as tarefas no código

- Situação prática que surge quando algoritmos paralelizados são usados em computadores com múltiplos processadores (multicore)



Fonte: https://faculty.idc.ac.il/gadi/PPTs/Chapter5-BarrierSynchronization.pptx