

Sistemas Hardware-Software

Aula 20 - Semáforos

Engenharia

Fabio Lubacheski

Maciel C. Vidal

Igor Montagner

Fábio Ayres

Conceito : Região Crítica

"Parte do programa que só pode ser rodada uma thread por vez"

- Elimina situações de concorrência
- Elimina também todo o paralelismo e pode se tornar gargalo de desempenho

Exemplo de Região Crítica

```
--  
12  double soma = 0;  
13  void *soma_parcial(void *_arg) {  
14      struct soma_parcial_args *spa = _arg;  
15  
16      for (int i = spa->start; i < spa->end; i++) {  
17          soma += spa->vetor[i];  
18      }  
19  
20      return NULL;  
21  }  
--
```

A variável **soma** é **compartilhada globalmente** e incrementada por duas ou mais threads ao mesmo tempo

Mutex - Semáforos Binários

Primitiva de sincronização para criação de regiões de exclusão mútua

- **Lock** – se estiver destravado, trava e continua
 - se não espera até alguém destravar
- **Unlock** – se tiver a trava, destrava
 - se não tiver retorna erro

Correção – última aula

Usando Mutex para sincronizar threads

1. Utilização da API pthreads para criar mutex
2. Entender quando usá-los e como diminuir seu custo

Problema produtor e consumidor

Duas threads compartilhar um buffer de tamanho limitado:

- O **produtor** insere itens no buffer.

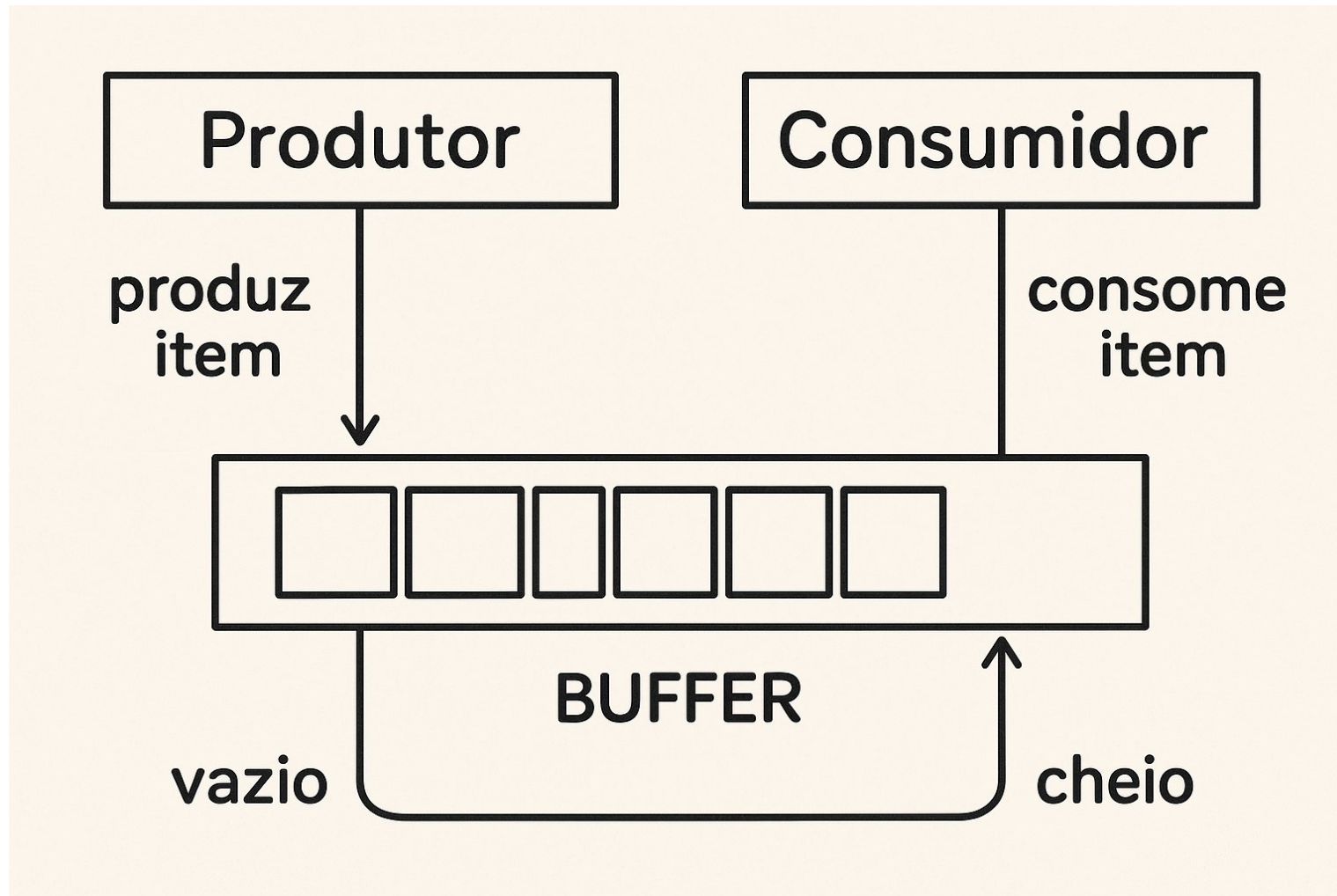
O **produtor** não produza quando o **buffer** estiver **cheio**

- O **consumidor** remove itens do buffer.

O consumidor **não consome** quando o buffer estiver vazio.

Ambos **não podem acessar o buffer ao mesmo tempo**, evitando **condições de corrida** (race conditions).

Problema – Produtor e Consumidor



Produtor/Consumidor (**pseudocódigo**)

```
produtor() {  
    while (true) {  
        item = produzir();  
  
        inserir(item);  
  
    }  
}
```

```
consumidor() {  
    while (true) {  
  
        item = retirar();  
  
        consumir(item);  
    }  
}
```


Semáforos contadores

"Inteiro especial que nunca fica negativo"

Só pode ser manipulado por duas operações atômicas

Post:

- Aumenta o valor em 1

Wait:

- Se for positivo, diminui em 1
- Se for 0 fica esperando;

Semáforos – Representação Conceitual

```
Wait(semaforo s)
{
    if( s > 0 )
        s = s - 1
    else
        "Bloqueia a thread"
}
```

```
Post(semaforo s)
{
    if("existe uma thread
        bloqueada" )
        "desbloqueia a
        thread"
    else
        s = s + 1
}
```

Produtor/Consumidor (pseudocódigo)

```
// semaforos compartilhados
```

```
Mutex      mutex = 1;
```

```
Semaforo cheio = 0;
```

```
Semaforo vazio = N; // tamanho do buffer
```

```
produtor() {  
    while (true) {  
        item = produzir();  
        Wait(vazio);  
        Lock(mutex);  
        inserir(item);  
        Unlock(mutex);  
        Post(cheio);  
    }  
}
```

```
consumidor() {  
    while (true) {  
        Wait(cheio);  
        Lock(mutex);  
        item = retirar();  
        Unlock(mutex);  
        Post(vazio);  
        consumir(item);  
    }  
}
```



Atividade prática

Semáforos no Papel (20 minutos)

1. Rendez-vous
2. Ordens de execução válida



Atividade prática

Semáforos POSIX (45 minutos)

1. Rendez-vous
2. Implementação usando semáforos e pthreads

Insper

www.insper.edu.br