# MAC0438 – Programação concorrente

Daniel Macêdo Batista

IME - USP, 12 de Abril de 2013

# Roteiro

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

**Problema dos leitores e escritores**

**Leitores e escritores com condição de sincronização**

**`await` com semáforos: A técnica de passagem de bastão**

Problema dos leitores e escritores
Leitores e escritores com condição de sincronização
`await` com semáforos: A técnica de passagem de bastão

# Problema dos leitores e escritores

# Descrição

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Dois tipos de processos (leitores e escritores) compartilham uma base de dados
- Leitores só examinam os registros
- Escritores examinam os registros e atualizam a base de dados
- Um escritor precisa de acesso exclusivo à base de dados
- Se nenhum escritor estiver acessando a base de dados, pode haver qualquer quantidade de leitores concorrentemente

# Descrição

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Exclusão mútua seletiva
  - Classes de processos competem pela base de dados
  - Leitores competem com escritores
  - Escritores competem entre si
- Leitores devem esperar até não haver nenhum escritor acessando a base
- Escritores devem esperar até não haver nenhum leitor ou outro escritor acessando a base

# Primeiro algoritmo

Problema dos leitores e escritores
Leitores e escritores com condição de sincronização
`await` com semáforos: A técnica de passagem de bastão

- Evitar que dois escritores acessem a base ao mesmo tempo
- Evitar que algum leitor acesse a base enquanto houver escritor acessando
- Começar pensando em uma solução mais geral pode ajudar

  Leitores e escritores são iguais e não podem acessar a base ao mesmo tempo

  **Como seria a solução?**

# Primeiro algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Resolve o problema da seção crítica
- Um único semáforo binário `rw` inicialmente valendo 1

```
sem rw = 1;

process Reader[i = 1 to M] {
   while (true) {
      ...
      P(rw);
      le a base;
      V(rw);
   }
}
```

# Primeiro algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

```
process Writer[j = 1 to N] {
   while (true) {
      ...
      P(rw);
      escreve na base;
      V(rw);
   }
}
```

# Primeiro algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Ineficiente pois leitores não compartilham a base
- **Como relaxar as restrições?**

  Leitores, como um grupo, precisam disputar o acesso com os escritores

  Leitores não precisam disputar o acesso entre si

# Primeiro algoritmo

Problema dos leitores e escritores
Leitores e escritores com condição de sincronização
`await` com semáforos: A técnica de passagem de bastão

- Apenas o primeiro leitor precisa disputar o acesso com os escritores
- Os demais leitores não precisam disputar acesso com ninguém
- Algum leitor só precisa liberar o acesso para os escritores se ele for o último leitor em execução
- **A verificação de se o leitor é o primeiro é uma seção crítica!**

# Primeiro algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

await com semáforos: A técnica de passagem de bastão

```
int nr = 0; /* Quantidade de leitores */
sem rw = 1;

process Reader [i=1 to M] {
   while (true) {
      ...
      <nr = nr+1;
       if (nr == 1) P(rw);
      >
      le a base;
      <nr = nr-1;
       if (nr == 0) V(rw);
      >
   }
}
```

# Primeiro algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

```
process Writer[j=1 to N] {
   ...
   P(rw);
   escreve na base;
   V(rw);
}
```

# Primeiro algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Mas estamos roubando com aquelas ações atômicas nos leitores!
- **Como tirar usando semáforos?**

```
...
<nr = nr+1;
 if (nr == 1) P(rw);
>
...
<nr = nr-1;
 if (nr == 0) V(rw);
>
```

# Primeiro algoritmo sem as ações atômicas explícitas

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

await com semáforos: A técnica de passagem de bastão

```
int nr = 0; /* Quantidade de leitores */
sem rw = 1;
sem mutexR = 1; /* Semaforo entre leitores */

process Reader [i=1 to M] {
   while (true) {
      ...
      P(mutexR);
         nr = nr+1;
         if (nr == 1) P(rw);
      V(mutexR);
      le a base;
      P(mutexR);
         nr = nr-1;
         if (nr == 0) V(rw);
      V(mutexR);
   }
}
```

# Primeiro algoritmo sem as ações atômicas explícitas

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

```
process Writer[j=1 to N] {
   ...
   P(rw);
   escreve na base;
   V(rw);
}
```

# Primeiro algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Simples
- **Problema?**
- Na segunda solução isso será resolvido

Problema dos leitores e escritores
Leitores e escritores com condição de sincronização
await com semáforos: A técnica de passagem de bastão

# Leitores e escritores com condição de sincronização

# Algoritmo anterior

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Escritores excluem uns aos outros
- Leitores, como classe, excluem escritores
- Dois protocolos de acesso à seção crítica

  Base de dados – `rw`

  Variável `nr` (quantidade de leitores) – `mutexR`

- Privilegia os leitores

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Relembrando a especificação do problema:

  Leitores examinam uma base de dados compartilhada

  Escritores examinam e alteram

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Pensando na quantidade de processos

  1 escritor requer acesso exclusivo

  Qualquer quantidade de leitores pode executar de forma concorrente

- Podemos usar condição de sincronização (`await`) a partir da quantidade de processos:

  Contamos a quantidade de cada tipo de processo

  Restringimos os valores

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Especificando o estado que queremos evitar

  `nr` e `nw` inteiros não negativos

  `nr`: # de leitores acessando a base

  `nw`: # de escritores acessando a base

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

```
/* Leitor */
<nr = nr+1;>
le a base;
<nr = nr-1;>

/* Escritor */
<nw = nw+1;>
escreve na base;
<nw = nw-1;>
```

- Mas falta especificar o que deve ser evitado
- Qual seria a frase para resumir o que queremos evitar?

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Queremos evitar que ambos `nr` **e** `nw` sejam maiores que zero **ou** que `nw` seja maior que 1.

- **Evitar**: `(nr > 0 && nw > 0) || nw > 1`

- é **garantir**: `(nr == 0 || nw == 0) && nw <= 1`

- Primeiro termo: leitores e escritores não podem acessar a base ao mesmo tempo

- Segundo termo: há no máximo um escritor ativo

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Lembrando o que já tínhamos:

```
/* Leitor */
<nr = nr+1;>
le a base;
<nr = nr-1;>

/* Escritor */
<nw = nw+1;>
escreve na base;
<nw = nw-1;>
```

- Precisamos incluir: `(nr == 0 || nw == 0) && nw <= 1`
- **A ação atômica incondicional tem que virar uma ação atômica condicional**

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

```
/* Leitor */
<nr = nr+1;>
le a base;
<nr = nr-1;>
```

- **garantir**: `(nr == 0 || nw == 0) && nw <= 1`
- Se `nr` vai ser incrementado, para garantir o primeiro termo, `nw` tem que ser quanto?

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

```
/* Leitor */
<await (nw == 0) nr = nr+1;>
le a base;
<nr = nr-1;>
```

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

```
/* Escritor */
<nw = nw+1;>
escreve na base;
<nw = nw-1;>
```

- **garantir**: `(nr == 0 || nw == 0) && nw <= 1`
- Se `nw` vai ser incrementado, para garantir o primeiro termo, `nr` tem que ser quanto?
- Se `nw` vai ser incrementado, para garantir o segundo termo, `nw` tem que ser quanto?

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

```
/* Escritor */
<await (nr == 0 && nw == 0) nw = nw+1;>
escreve na base;
<nw = nw-1;>
```

# Segundo algoritmo

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

await com semáforos: A técnica de passagem de bastão

```
int nr = 0, nw = 0;

process Reader [i = 1 to m] {
   while (true) {
      ...
      <await (nw == 0) nr = nr+1;>
      le a base;
      <nr = nr-1;>
   }
}
```

# Segundo algoritmo

Problema dos leitores e escritores
Leitores e escritores com condição de sincronização
await com semáforos: A técnica de passagem de bastão

```
process Writer [j = 1 to n] {
   while (true) {
      ...
      <await (nr == 0 and nw == 0) nw = nw+1;>
      escreve na base;
      <nw = nw-1;>
   }
}
```

- Ok, mas temos que tirar os await!

Problema dos leitores e escritores
Leitores e escritores com condição de sincronização
await com semáforos: A técnica de passagem de bastão

# `await` com semáforos: A técnica de passagem de bastão

# `await` e semáforos

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- É difícil substituir os `await` por semáforos
- No caso anterior:

```
<await (nr == 0 and nw == 0) ...;>

<await (nw == 0) ...;>
```

- As condições não são as mesmas. Não tem como simplesmente trocar o await por um semáforo (Lembrando que as operações do semáforo são sobre uma única variável)

# `await` e semáforos

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- A técnica de passagem de bastão permite implementar `await` com semáforos!

  Serve para as condições de sincronização completas (`<await (B) S;>`). Lembrando que `<S;>` pode ser escrita como `<await (true) S;>`

  Podemos resolver qualquer problema com condição de sincronização!

# Usando semáforos binários divididos para o `await`

Problema dos leitores e escritores

Leitores e escritores com condição de sincronização

`await` com semáforos: A técnica de passagem de bastão

- Usamos para o problema do produtor/consumidor com um buffer simples (`full` e `empty` sempre somavam 1)
- Considere:

  e: semáforo binário, inicialmente valendo 1. Controla a entrada em cada ação atômica

  1 semáforo e 1 contador associados a cada expressão `B`, inicialmente ambos valendo 0. O semáforo fará os processos esperarem `B` ser verdade. O contador contará o número de processos esperando `B` ser verdade.

# Aplicando a técnica no problema dos leitores e escritores

Problema dos leitores e escritores
Leitores e escritores com condição de sincronização
`await` com semáforos: A técnica de passagem de bastão

- Escritor: `<await (nr == 0 and nw == 0) ...;>`
- Leitor: `<await (nw == 0) ...;>`

- Semáforos:

  `e`: semáforo que controla a entrada em cada ação atômica

  `r`: semáforo associado com a condição do leitor

  `dr`: número de leitores esperando a condição ser verdade

  `w`: semáforo associado com a condição do escritor

  `dw`: número de escritores esperando a condição ser verdade