
Condições de Disputa

Volnys Borges Bernal
volnys@lsi.usp.br

Departamento de Sistemas Eletrônicos
Escola Politécnica da USP



Agenda

- ❑ **Condições de disputa**
- ❑ **Região Crítica**

Sobre esta apresentação

- **Esta apresentação ...**
 - ❖ Não apresenta todos os detalhes sobre este tópico.
 - ❖ É um resumo para auxiliar a apresentação do tópico em sala de aula.

- **Para estudo, deve ser utilizada uma das seguintes referências:**
 - ❖ **Capítulos 1 e 2 do livro:**
 - ANDREW S. TANENBAUM; Sistemas Operacionais Modernos. Prentice-Hall
 - ❖ **Capítulos 1 e 2 do livro:**
 - ANDREW S. TANENBAUM; Sistemas Operacionais. Prentice-Hall.

Condições de disputa (*Race Conditions*)



Condições de disputa

- ❑ **Condição de disputa é**
 - ❖ Uma situação de conflito ...
 - ❖ No acesso a um determinado recurso (variável, estrutura, arquivo, ...)
 - ❖ Recurso este compartilhado
 - ❖ Por duas ou mais entidades de processamento (processos, *threads*, ...)
 - ❖ Que pode causar resultados não esperados

- ❑ **Importante:**
 - ❖ **Threads de um mesmo processo possuem diversos recursos compartilhados**
 - Área de dados
 - Arquivos abertos
 - etc
 - ❖ **Quando existem acessos de escrita a estes recursos compartilhados podem ocorrer potenciais situações de condição de disputa**

Condições de disputa

- **A seguir, serão apresentados 3 exemplos de condição de disputa:**
 - ❖ **Exemplo 1: Contador**
 - ❖ **Exemplo 2: Manipulação de lista ligada**
 - ❖ **Exemplo 3: Variável de proteção**

Exemplo 1: Contador



Exemplo 1: Contador

□ Descrição

- ❖ Dois *threads* realizam determinadas tarefas.
- ❖ Após realizar cada tarefa incrementam um contador *c*.
- ❖ Variável *c* é global (compartilhada entre os dois threads)

Thread1:

```
...  
repetir  
    <Realiza tarefa>  
    c = c + 1  
...
```

Thread2:

```
...  
repetir  
    <Realiza tarefa>  
    c = c + 1  
...
```

Exemplo 1: Contador

❑ Versão do programa em assembler

Thread1 :

```
...  
repete: ...  
        realiza tarefa  
...  
LOAD    AC, (c)  
ADD     AC, 1  
STORE   (c), AC  
JUMP    repete  
...
```

Thread2 :

```
...  
repete: ...  
        realiza tarefa  
...  
LOAD    AC, (c)  
ADD     AC, 1  
STORE   (c), AC  
JUMP    repete  
...
```

Exemplo 1: Contador

□ Problema: condição de disputa sobre o contador “c”

❖ Em sistemas monoprocessoadores

- Concorrência:

- troca de contexto durante a atualização do contador “c”

❖ Em sistemas multiprocessoadores

- Concorrência:

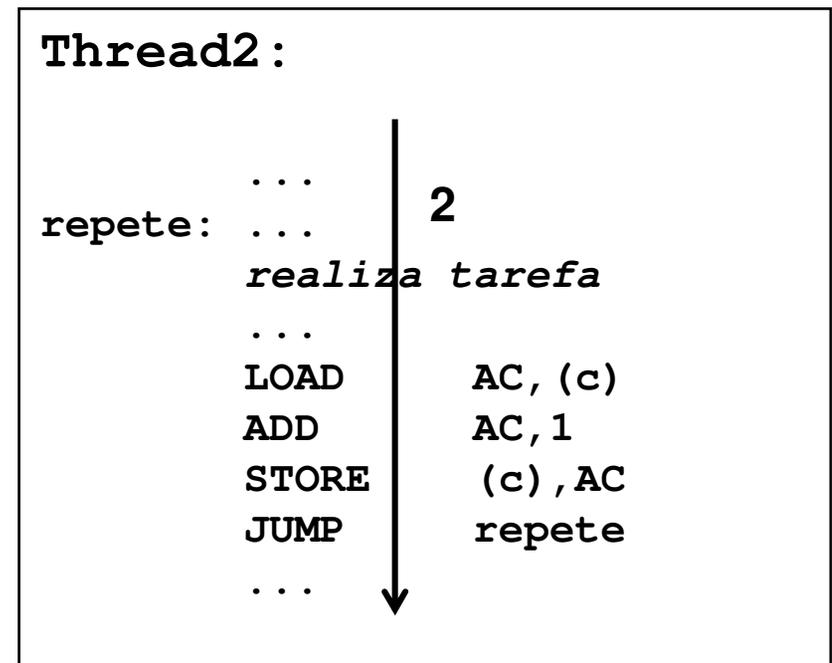
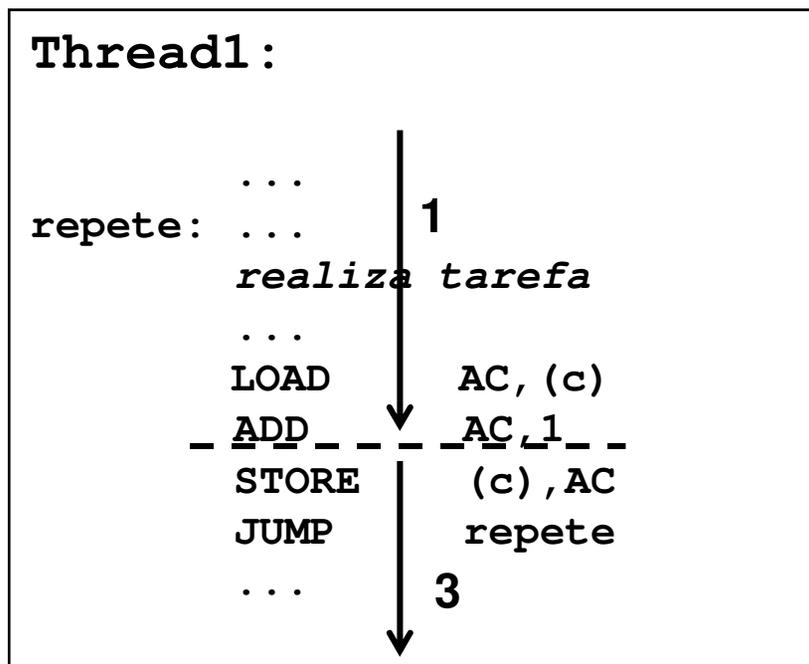
- troca de contexto durante a atualização do contador “c”

- Paralelismo:

- incremento simultâneo do contador “c”

Exemplo 1: Contador

- Condição de disputa na concorrência:



Exemplo 1: Contador

□ Condição de disputa no paralelismo

Thread1 :

```
...  
repete: ...  
        realiza tarefa  
...  
LOAD   ↓1 AC, (c)  
ADD    ↓2 AC, 1  
STORE  ↓3 (c), AC  
JUMP   repete  
...
```

Thread2 :

```
...  
repete: ...  
        realiza tarefa  
...  
LOAD   ↓1 AC, (c)  
ADD    ↓2 AC, 1  
STORE  ↓3 (c), AC  
JUMP   repete  
...
```

Exemplo 2: Lista ligada



Exemplo 2: Lista ligada

□ Exemplo 2: Manipulação de lista ligada

- ❖ Quando dois ou mais *threads* manipulam uma lista ligada, com pelo menos um thread modificando a lista ligada.

Thread1:

...

<manipula lista ligada>

...

Thread2:

...

<manipula lista ligada>

...

Exemplo 2: Lista ligada

- ❑ **Problema: condição de disputa durante a manipulação da lista ligada:**
 - ❖ **Em sistemas monoprocessoadores**
 - Concorrência: A troca de contexto durante a modificação da lista pode deixa-la em um estado inconsistente

 - ❖ **Em sistemas multiprocessoadores**
 - Concorrência:
 - A troca de contexto durante a modificação da lista pode deixa-la em um estado inconsistente
 - Paralelismo:
 - A modificação da lista por um dos threads pode deixa-la em um estado inconsistente.

Exemplo 3: Variável de proteção



Exemplo 3: Variável de proteção

□ Descrição

- ❖ Dois *threads* definem uma variável compartilhada para controle do uso do recurso.
- ❖ Se a variável for 1 significa que o recurso está ocupado, se for zero está livre.

Thread1:

```
...  
while (ocupado == 1);  
ocupado = 1;  
<usa recurso>  
ocupado = 0;  
...
```

Thread2:

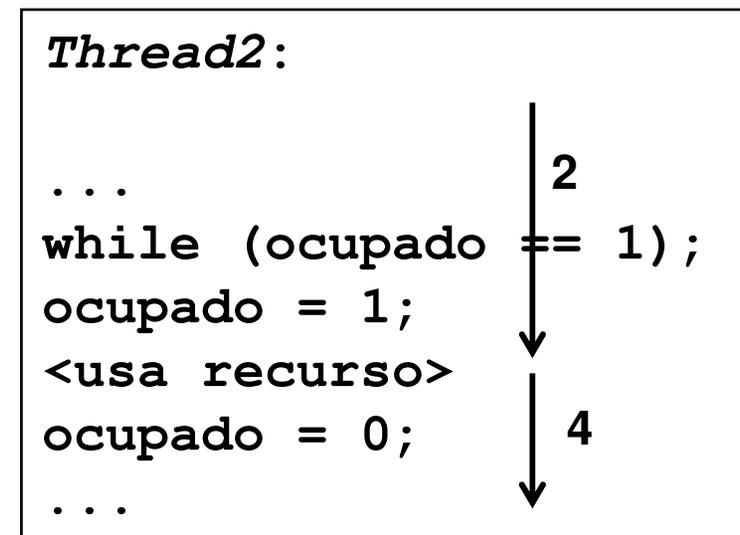
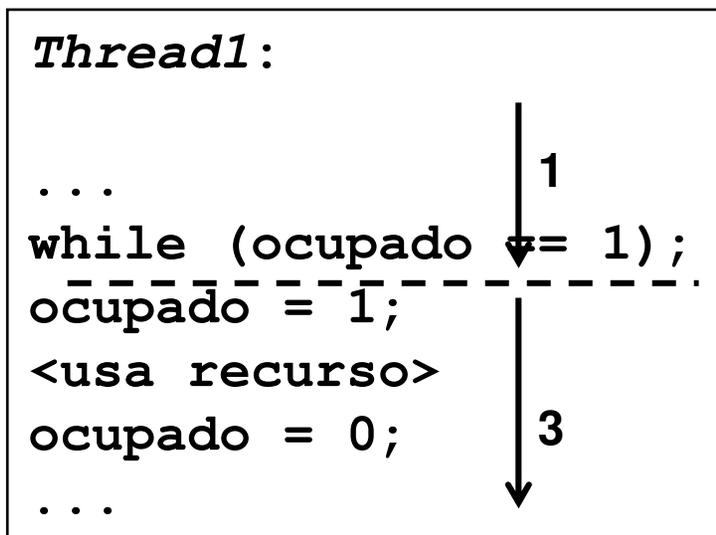
```
...  
while (ocupado == 1);  
ocupado = 1;  
<usa recurso>  
ocupado = 0;  
...
```

Exemplo 3: Variável de proteção

- **Problema: Condição de disputa sobre a variável “ocupado”**
 - ❖ **Em sistemas monoprocessadores**
 - Concorrência:
 - Troca de contexto durante a alteração da variável “ocupado” para 1
 - ❖ **Em sistemas multiprocessadores**
 - Concorrência:
 - Troca de contexto durante a alteração da variável “ocupado” para 1
 - Paralelismo:
 - Dois *threads* alterando simultaneamente a variável “ocupado” para 1

Exemplo 3: Variável de proteção

- Condição de disputa quando existe concorrência



Exemplo 3: Variável de proteção

- Condição de disputa quando existe paralelismo

Thread1:

```
...  
while (ocupado == 1); ↓1  
ocupado = 1;           ↓2  
<usa recurso>  
ocupado = 0;  
...
```

Thread2:

```
...  
while (ocupado == 1); ↓1  
ocupado = 1;           ↓2  
<usa recurso>  
ocupado = 0;  
...
```

Região Crítica



Região Crítica

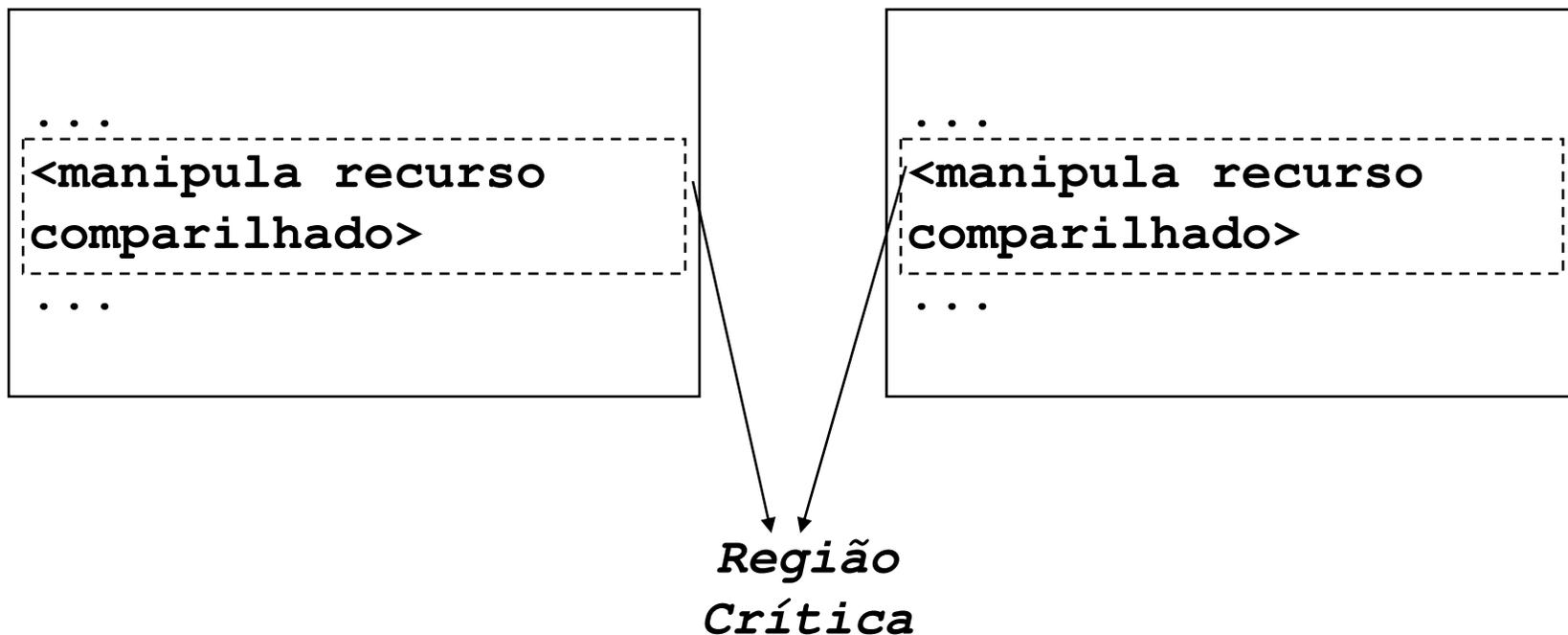
- **Região crítica é ...**
 - ❖ Uma região de código ...
 - ❖ Na qual existe acesso a recursos compartilhados ...
 - ❖ Que resulta em condição de disputa

- **Objetivo da região crítica**
 - ❖ Identificar a região de código na qual existe potencialmente ocorrência de condição de disputa devido ao acesso por duas ou mais entidades
 - ❖ Possibilitar a utilização de soluções de sincronização para evitar condição de disputa na região crítica

- **Obs:**
 - ❖ Entidades
 - Processos, threads,

Região Crítica

- ❑ Exemplo de região de código na qual existe acesso a recursos compartilhados que pode causar problema de condição de disputa

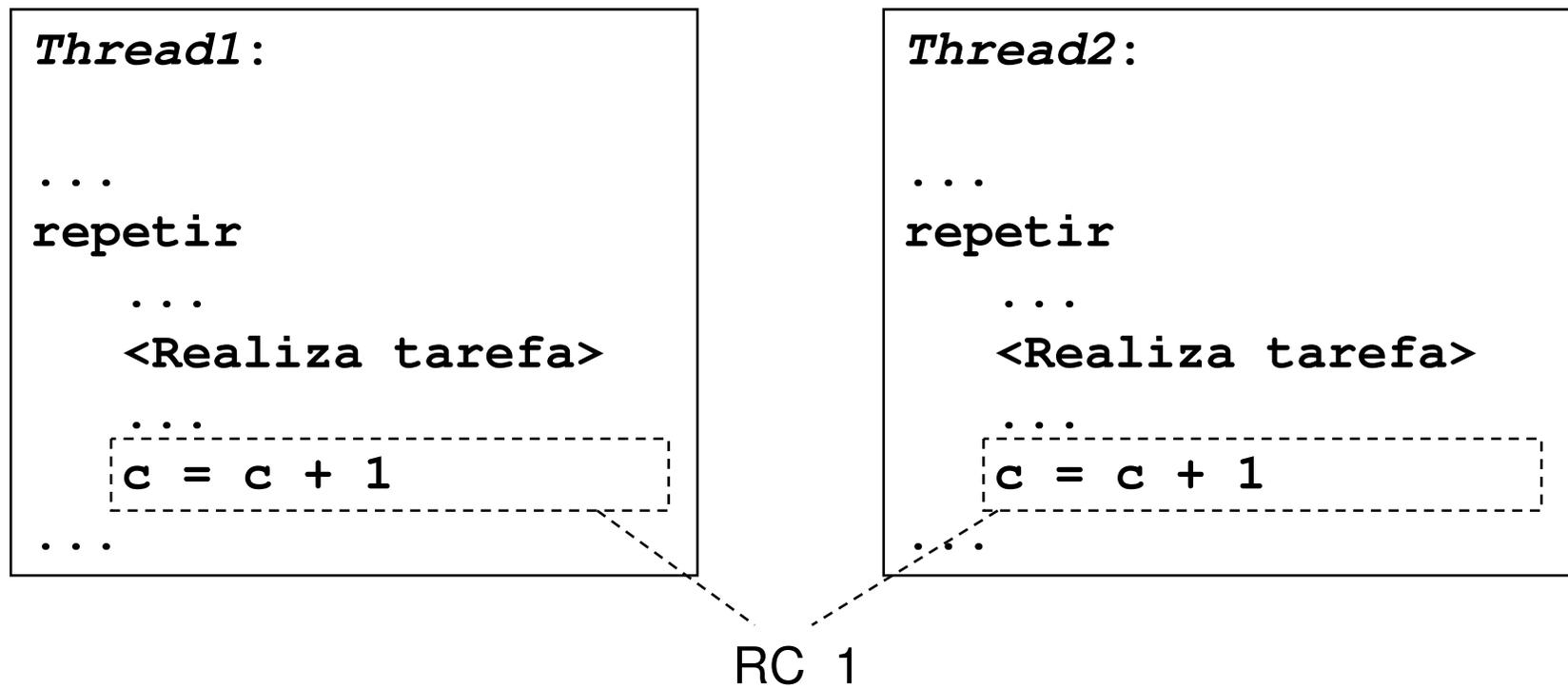


Região Crítica

□ Exemplo 1:

❖ Contador

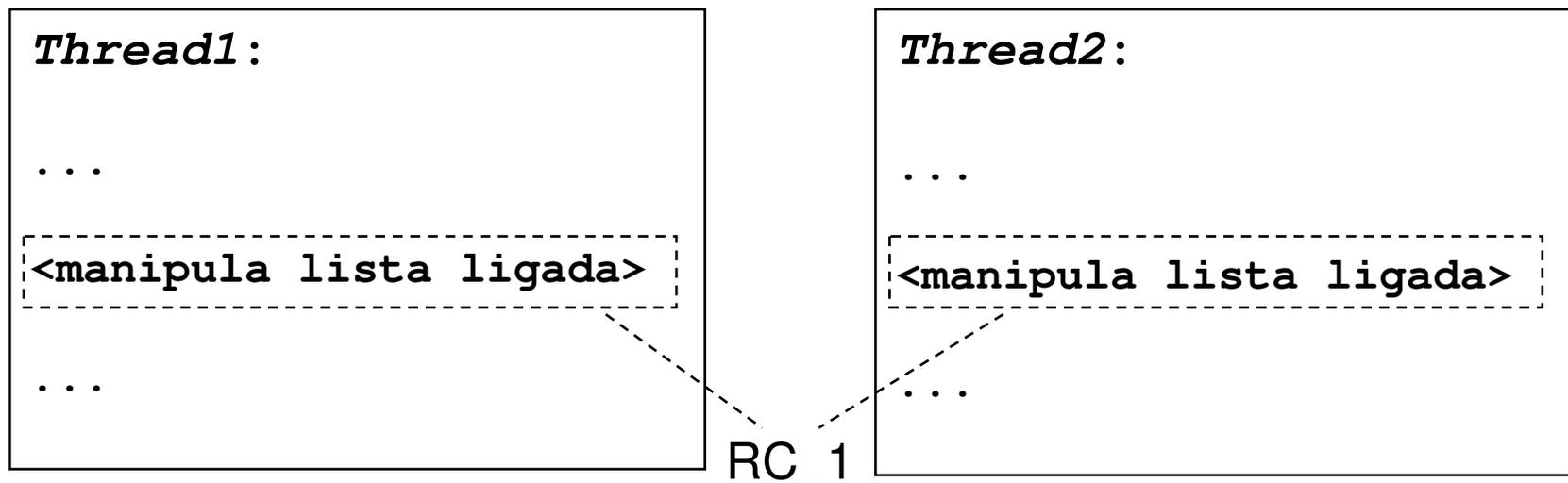
- Contador c compartilhado (variável global)



Região Crítica

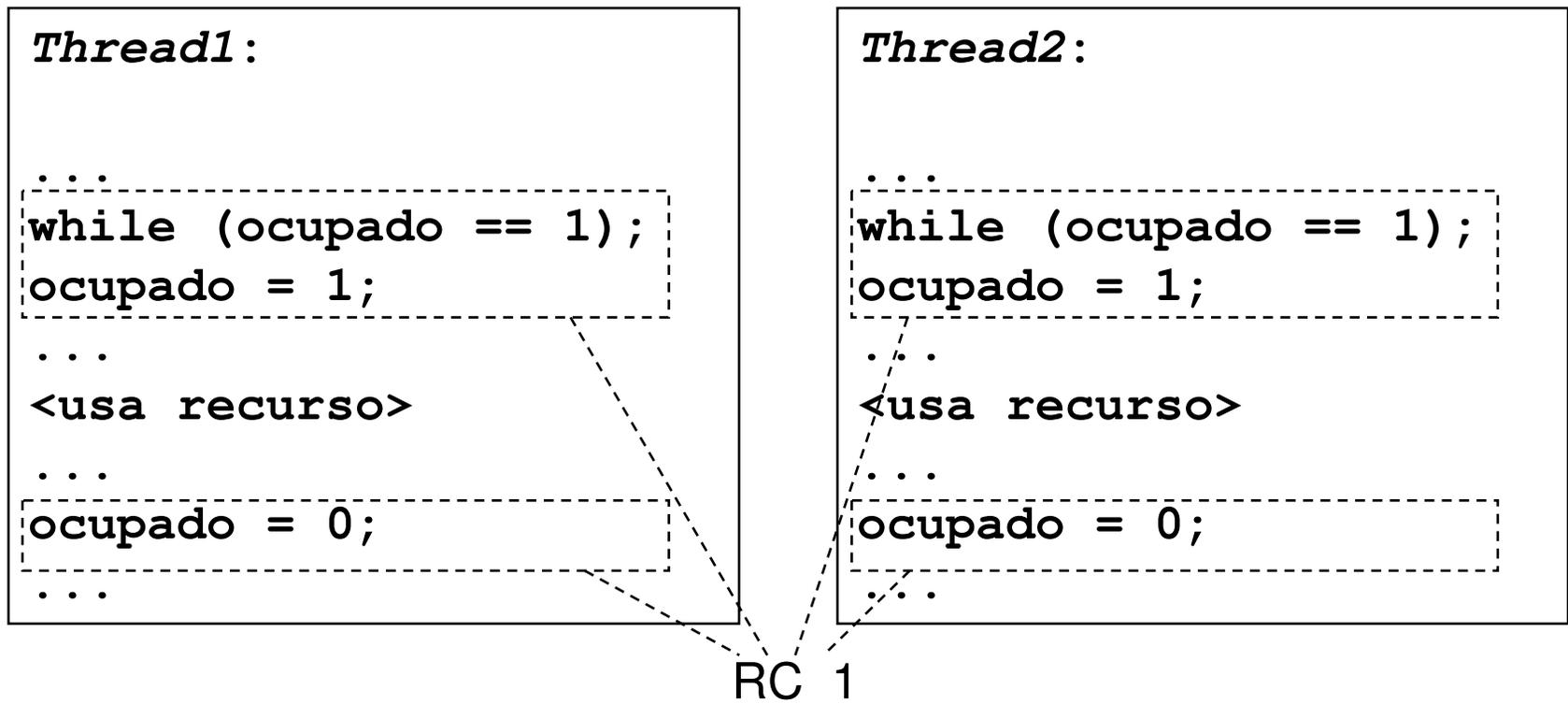
□ Exemplo 2:

❖ Manipulação de lista ligada compartilhada



Região Crítica

- Exemplo 3:
 - ❖ Variável de proteção



Referências Bibliográficas



Referências Bibliográficas

- **ANDREW S. TANENBAUM; Sistemas Operacionais Modernos. Prentice-Hall.**
 - ❖ **Capítulo 2**

- **ANDREW S. TANENBAUM; Sistemas Operacionais. Prentice-Hall.**
 - ❖ **Capítulo 2**