centro universitário FEI

Lab: Sincronização de Processos

Sistemas Operacionais

Prof. Charles Ferreira
cferreira@fei.edu.br

Agenda

Sincronização de processos

Exercícios

Lab: Sincronização de Processos

Sincronização de processos

Como poderíamos resolver o problema do produtor consumidor de buffer limitado

?

Sincronização de processos

Para resolver o problema precisamos garantir que:

- A região crítica seja acessada por uma única thread por vez.
- A thread que chegar primeiro irá adquirir um “bloqueio”
- a qual irá garantir que somente ela irá acessar aquela área.

Sincronização de processos

Bloqueio na região crítica

- Caso a thread que esteja na região crítica não posso trabalhar …
  - então ela deve dormir e liberar o bloqueio.
- Depois que a thread que estiver na região crítica acabar …
  - ela deve acordar a thread que está dormindo.

Implementação

Sincronização de métodos

- Em java podemos utilizar a palavra reservada **synchronized**
- Métodos synchronized só podem ser acessados por uma única thread.
- Garantindo a exclusão mútua.

Espera ocupada

```
while(true){
  ;
}
```

O problema da espera ocupada

- Outro problema da implementação inicial é a espera ocupada.
- Ela desperdiça ciclos de CPU.
- Idealmente, as threads devem “dormir” enquanto não podem trabalhar.

Wait and notify

Comando wait()

- Faz a thread dormir.

Comando notify()

- Acorda a thread que estiver dormindo.

Resolução

Vamos adaptar o programa anterior para que:

- Aconteça sincronização entre os processos.
- Não tenha espera ocupada.

BoundedBuffer

Método insert

```
public synchronized void insert(int item) {
  while (contador == BUFFER_SIZE) {
    try {
      wait();
    }catch (InterruptedException e) {
    }
  }
  buffer[in] = item;
  System.out.println("Produzido: contador: " + contador);
  in = (in + 1) % BUFFER_SIZE;
  contador++;
  notify();
}
```

BoundedBuffer

Método remove

```
public synchronized int remove() {
  while (contador == 0) {
    try {
      wait();
    }catch (InterruptedException e) {
    }
  }
  int item = buffer[out];
  System.out.println("Consumido: contador: " + contador);
  out = (out + 1) % BUFFER_SIZE;
  contador--;
  notify();
  return item;
}
```

Lab: Sincronização de Processos

Exercícios

Exercício 1

Implemente um sistema Produtor Consumidor com sincronização

- Deverá haver:
  - BoundedBuffer contendo um **Array de Strings**
  - 2 threads produtoras produzindo **Strings**.
  - 3 threads consumidoras consumindo **Strings**.
    - elas devem imprimir o conteúdo retirado do buffer
- Conteúdo a ser produzido/consumido:
  - A data em que a string foi produzida junto com o ID da thread produtora

Adicione o atributo ID na thread Produtora

Exercício 1

Exemplos de dados produzidos e inseridos no vetor

Classe com método main

```java
public class Fabrica {

  public static void main(String[] args) {
    BoundedBuffer buffer = new BoundedBuffer();

    Thread produtora1 = new Thread(new Produtor(buffer, 1));
    Thread produtora2 = new Thread(new Produtor(buffer, 2));
    Thread consumidora1 = new Thread(new Consumidor(buffer));
    Thread consumidora2 = new Thread(new Consumidor(buffer));
    Thread consumidora3 = new Thread(new Consumidor(buffer));

    produtora1.start();
    produtora2.start();
    consumidora1.start();
    consumidora2.start();
    consumidora3.start();
  }
}
```

Saída esperada

```
Produzi: buffer: 0
Consumi: TID 1: Thu Sep 22 10:54:29 BRT 2022 buffer: 0
Produzi: buffer: 0
Consumi: TID 1: Thu Sep 22 10:54:29 BRT 2022 buffer: 0
Produzi: buffer: 0
Consumi: TID 1: Thu Sep 22 10:54:30 BRT 2022 buffer: 0
Produzi: buffer: 0
Consumi: TID 2: Thu Sep 22 10:54:29 BRT 2022 buffer: 0
Produzi: buffer: 0
Consumi: TID 2: Thu Sep 22 10:54:33 BRT 2022 buffer: 0
Produzi: buffer: 0
Consumi: TID 2: Thu Sep 22 10:54:34 BRT 2022 buffer: 0
Produzi: buffer: 0
Consumi: TID 2: Thu Sep 22 10:54:36 BRT 2022 buffer: 0
Produzi: buffer: 0
Consumi: TID 1: Thu Sep 22 10:54:32 BRT 2022 buffer: 0
```

Obrigado

cferreira@fei.edu.br