Produtor Consumidor

Axell Brendow Batista Moreira

[1] Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Caixa Postal 1.686 – 30.535-901 – Belo Horizonte – MG – Brazil

Uma ou mais thread de produtores criam um produto e colocam em um buffer. Uma ou mais thread de consumidores consomem o produto colocado no buffer. O produtor precisa esperar o buffer ficar livre para produzir o produto e o cliente precisa esperar o buffer ficar preenchido para consumir o produto.

A tarefa do problema é sincronizar o acesso ao recurso, no caso a pilha, para que produtores saibam quando podem produzir e consumidores saibam quando podem consumir.

Produtor Consumidor – Implementação em Java

```
class Buffer {

   private int conteudo;
   private boolean disponivel;

   public synchronized void set(int idProdutor, int valor) {
      while (disponivel == true) {
         try {
            System.out.println("Produtor #" + idProdutor + "
               esperando...");
            wait();
         } catch (Exception e) {
            e.printStackTrace();
         }
      }
      conteudo = valor;
      System.out.println("Produtor #" + idProdutor + " colocou
         " + conteudo);
      disponivel = true;
      notifyAll();
   }

   public synchronized int get(int idConsumidor) {
      while (disponivel == false) {
         try {
            System.out.println("Consumidor #" + idConsumidor
                  + " esperado...");
            wait();
```

```
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        System.out.println("Consumidor #" + idConsumidor + "
            consumiu: "
                + conteudo);
        disponivel = false;
        notifyAll();
        return conteudo;
    }
}

class Produtor extends Thread {
    private int idProdutor;
    private Buffer pilha;
    private int producaoTotal;

    public Produtor(int id, Buffer p, int producaoTotal) {
        idProdutor = id;
        pilha = p;
        this.producaoTotal = producaoTotal;
    }

    public void run() {
        for (int i = 0; i < producaoTotal; i++) {
            pilha.set(idProdutor, i);
        }
        System.out.println("Produtor #" + idProdutor + "
            concluido!");
    }
}

class Consumidor extends Thread {
    private int idConsumidor;
    private Buffer pilha;
    private int totalConsumir;

    public Consumidor(int id, Buffer p, int totalConsumir) {
        idConsumidor = id;
        pilha = p;
        this.totalConsumir = totalConsumir;
    }

    public void run() {
        for (int i = 0; i < totalConsumir; i++) {
            pilha.get(idConsumidor);
        }
        System.out.println("Consumidor #" + idConsumidor + "
```

```
            concluido!");
    }
}

public class consumidorProdutor
{
    public static void main(String[] args) {
        Buffer bufferCompartilhado = new Buffer();
        Produtor produtor1 = new Produtor(1, bufferCompartilhado,
            5);
        Produtor produtor2 = new Produtor(2, bufferCompartilhado,
            5);
        Consumidor consumidor1 = new Consumidor(1,
            bufferCompartilhado, 2);
        Consumidor consumidor2 = new Consumidor(2,
            bufferCompartilhado, 8);

        produtor1.start();
        consumidor1.start();
        produtor2.start();
        consumidor2.start();
    }
}
```

Produtor Consumidor – Implementação em C

```
// descricao: Programa produtor-consumidor com mutex
// Utiliza a biblioteca pthreads.
// para compilar: cc -o phtread pthread.c -lpthread

#include <pthread.h>

#define FALSE 0
#define TRUE 1


// Declaracao das variaveis de condicao:
pthread_mutex_t mutex;

//Buffer
#define BUFFERVAZIO 0
#define BUFFERCHEIO 1
int buffer;
int estado = BUFFERVAZIO;


void produtor(int id)
      {
```

```
   int i=0;
   int item;
      int aguardar;

   printf("Inicio produtor %d \n",id);
   while (i < 10)
      {
         //produzir item
      item = i + (id*1000);

         do
         {
            pthread_mutex_lock(&mutex);
         aguardar = FALSE;
         if (estado == BUFFERCHEIO)
         {
         aguardar = TRUE;
            pthread_mutex_unlock(&mutex);
         }
      } while (aguardar == TRUE);

      //inserir item
         printf("Produtor %d inserindo item %d\n", id, item);
         buffer = item;
      estado = BUFFERCHEIO;

      pthread_mutex_unlock(&mutex);
      i++;
      sleep(2);
      }
   printf("Produtor %d terminado \n", id);
      }

void consumidor(int id)
      {
   int item;
   int aguardar;

   printf("Inicio consumidor %d \n",id);
   while (1)
      {
         // retirar item da fila
         do
      {
            pthread_mutex_lock(&mutex);
         aguardar = FALSE;
         if (estado == BUFFERVAZIO)
         {
            aguardar = TRUE;
```

```
            pthread_mutex_unlock(&mutex);
            }
            } while (aguardar == TRUE);
        item = buffer;
        estado = BUFFERVAZIO;
        pthread_mutex_unlock(&mutex);

        // processar item
            printf("Consumidor %d consumiu item %d\n", id, item);

        sleep(2);
        }
    printf("Consumidor %d terminado \n", id);
        }

int main()
{
    pthread_t prod1;
    pthread_t prod2;
    pthread_t prod3;
    pthread_t cons1;
    pthread_t cons2;

    printf("Programa Produtor-Consumidor\n");

    printf("Iniciando variaveis de sincronizacao.\n");
    pthread_mutex_init(&mutex,NULL);

        printf("Disparando threads produtores\n");
    pthread_create(&prod1, NULL, (void*) produtor,1);
    pthread_create(&prod2, NULL, (void*) produtor,2);
    pthread_create(&prod3, NULL, (void*) produtor,3);

        printf("Disparando threads consumidores\n");
    pthread_create(&cons1, NULL, (void*) consumidor,1);
    pthread_create(&cons2, NULL, (void*) consumidor,2);

    pthread_join(prod1,NULL);
    pthread_join(prod2,NULL);
    pthread_join(prod3,NULL);
    pthread_join(cons1,NULL);
    pthread_join(cons2,NULL);

        printf("Terminado processo Produtor-Consumidor.\n\n");
}
```