Atividade - Aula 19/08

Atenção: Vale ressaltar que esta atividade será usada como critério para a contabilização de sua frequência de aula.

Prazo de Entrega: 23/08/2021

Aluno: Pedro Vinícius da Silva Ribeiro Mat:2019033903

1. Com relação ao **Problema do Produtor/Consumidor** que consiste em dois processos compartilham um buffer de tamanho fixo. O processo produtor coloca dados no buffer e o processo consumidor retira dados do buffer, apresente:

A) A implementação de um programa, na linguagem C, que execute o problema do produtor/consumidor com um buffer de tamanho 10, usando primitivas sleep/wakeup.

B) Descreva o problema relacionado ao uso de primitivas sleep/wakeup em relação ao produtor/consumidor.

- Dois ou mais processos não podem estar simultaneamente dentro de suas regiões críticas correspondentes.
- Nenhuma consideração pode ser feita a respeito da velocidade relativa dos processos, ou a respeito do número de processadores disponíveis no sistema.
- Nenhum processo que esteja rodando fora de sua região crítica pode bloquear a execução de outro processo.
- Nenhum processo pode ser obrigado a esperar indefinidamente para entrar em sua região crítica.

C) Apresente uma solução extra para a implementação do problema do produtor/consumidor. Dica: Pesquise por mutex.

```c
//Resolucao usando mutex
// arquivo: prodcons.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");
}

```

```c
// descricao: Programa produtor-consumidor
// Utiliza a biblioteca pthreads.

#include <stdio.h>
#include <pthread.h>

int sb;
int turn = 0;

void put(int i){
    sb = i;
}

void *producer(){
    int i = 0;
    while(1){
        printf(">> Producer \n");
        while(turn == 1);
        printf("\t - Turn Producer \n");
        put(i);
        turn = 1;
        i = i + 1;
    }
}

int get(){
    return sb;
}

void *consumer(){
    int i,v;
    while(1){
        printf(">> Consumer \n");
        while(turn == 0);
        printf("\t - Turn Consumer \n");
        v = get();
        turn = 0;
        printf("\t - Peguei o valor: %d\n", v);
    }
}

int main(){
    pthread_t producer_t;
    pthread_t consumer_t;

    pthread_create(&producer_t, NULL, producer, NULL);
    pthread_create(&consumer_t, NULL, consumer, NULL);

    pthread_join(producer_t, NULL);
    pthread_join(consumer_t, NULL);


    return 0;
}
```