Laboratório 6
Semáforos e o padrão produtor/consumidor

Programação Concorrente (ICP-361)
Prof. Silvana Rossetto

[1]Instituto de Computação/UFRJ

Introdução

O objetivo deste Laboratório é introduzir e praticar o uso de semáforos para implementar exclusão mútua e sincronização condicional em programas concorrentes.

Atividade 1

Objetivo: Mostrar uma implementação do padrão produtor/consumidor usando semáforos.

Roteiro: Abra o arquivo **pc.c** (aula do dia 14/5) e siga os passos abaixo:

1. Leia o programa para entender como ele funciona.
2. Qual é a finalidade das variáveis **in** e **out** e por que foram definidas como variáveis **static**?
3. Execute o programa **várias vezes** e observe os resultados impressos na tela. (O programa executa em loop infinito, então sua execução precisará ser interrompida.)
4. O programa executou corretamente? Como isso foi demonstrado?

Atividade 2

Objetivo: Projetar e implementar um programa concorrente em C que usa o padrão produtor/consumidor.

Descrição: Vamos retomar com o problema da primalidade :-)

Implemente um programa concorrente onde UMA thread PRODUTORA carrega de um arquivo binário uma sequência de N (N bastante grande) de **números inteiros** e os deposita em um **buffer** de tamanho M (M pequeno) — um de cada vez — que será compartilhado com threads CONSUMIDORAS. AS threads CONSUMIDORAS retiram os números — um de cada vez — e avaliam sua primalidade, usando a seguinte função:

```
int ehPrimo(long long int n) {
   int i;
      if (n<=1) return 0;
      if (n==2) return 1;
      if (n%2==0) return 0;
      for (i=3; i<sqrt(n)+1; i+=2)
         if(n%i==0) return 0;
      return 1;
```

Ao final do programa (depois que os N números foram processados), deverá ser retornado: (i) a quantidade de números primos encontrados (para avaliar a corretude do programa); e (ii) a thread consumidora VENCEDORA (aquela que encontrou a maior quantidade de primos).

Roteiro:

1. Comece gerando os arquivos (binários) de teste, com N valores inteiros positivos e a quantidade de primos.
2. Implemente o programa que deve receber como entrada: a quantidade de threads consumidoras; o tamanho M do buffer e o nome do arquivo de entrada.
3. Verifique se o resultado final está correto (quantidade de primos encontrados).
4. Execute o programa várias vezes, alterando os parâmetros de entrada e ateste sua corretude.
5. Não se esqueça de garantir a organização e modularização do código.

Entrega do laboratório Disponibilize o código implementado na **Atividade** 2 em um ambiente de acesso remoto (GitHub ou GitLab). Use o formulário de entrega desse laboratório para enviar o link do repositório do código implementado e acrescentar comentários ou dúvidas.