Unifesp - ICT - BCT/BCC/EC
Programação Concorrente e Distribuída - PCD
Profs. Álvaro e Denise

Atividades de Programação - 2

Instruções:

- As atividades poderão ser entregues em grupo de até 4 integrantes.
- Entrega:
    - link no GitHub (ou similar) com o código desenvolvido
    - link para os vídeos descrevendo o código e seu funcionamento (máx. 5 min)
        - 1 vídeo por atividade
        - usar drive institucional ou Youtube
        - mostrar: principais trechos do código, exemplos de funcionamento (programa em execução), resultados obtidos (comentários, gráficos quando se aplicar)
        - sugestão: usar slides para informações estáticas, sem esquecer de demonstrar o programa em execução

  OBS: os links podem ser entregues via arquivo no Google Docs

Atividade 1 - Seção Crítica por espera ocupada

Implemente, usando linguagem C com PThreads ou OpenMP ou ainda em JavaThreads, o algoritmo de **Manna-Pnueli** que implementa entrada em SC por algoritmo Cliente-Servidor. Demonstre o funcionamento do código para 2 e 4 threads como processos clientes, realizando o correto incremento em uma determinada variável global ou ainda através de "prints" que demonstrem o funcionamento correto da Exclusão Mútua para a seção crítica.
OBS: Para provar que a seção crítica no código implementado de fato funciona, pode-se fazer um teste em dois códigos, onde o primeiro não implementa controle de seção crítica nos processos clientes (comentar o pré-protocolo, por exemplo) e o segundo faz a correta implementação da seção crítica nos processos clientes, e comparar as duas versões. Nestes testes faça com que a seção crítica seja executada inúmeras vezes pelos processos clientes (normalmente da ordem de bilhão de vezes). Segue abaixo um exemplo de seção crítica que pode ser usada no problema.

Considerando a variável SOMA como global/compartilhada, tem-se o seguinte trecho de código:

```
{
  int local = SOMA;
  sleep(rand()%2);
  SOMA = local + 1;
}
```

Algorithm 5.9: Manna-Pnueli central server algorithm	
integer request ← 0, respond ← 0	
client process i	
	loop forever
	non-critical section
p1:	while respond ≠ i
p2:	request ← i
	critical section
p3:	respond ← 0
server process	
	loop forever
p4:	await request ≠ 0
p5:	respond ← request
p6:	await respond = 0
p7:	request ← 0

Atividade 2 - Somatórias, seção crítica e reduções em OpenMP

A partir do programa Jogo da Vida já desenvolvido em atividade de programação anterior, a partir da versão desenvolvida em linguagem C/C++ e OpenMP, modifique o procedimento/função (ou trecho de código) que realiza a somatória de todas as posições da **última** geração do tabuleiro (soma a quantidade total de células vivas no tabuleiro), desenvolvendo duas versões:

a) Utilizar a diretiva `#pragma omp critical` para realizar a operação entre as *threads*, totalizando os resultados em uma variável global ao final;

b) Utilize uma operação de redução através da diretiva `#pragma omp for reduction(???)` para realizar a mesma operação.

c) Crie uma versão desse trecho do código onde não se pode utilizar nenhuma das soluções acima (itens *a* e *b*), ou outro recurso do OpenMP baseado em diretivas e cláusulas para realizar a somatória. É proibido também utilizar semáforos. Pode-se apenas utilizar recursos usuais de programação da linguagem C/C++.

Posteriormente, verifique o desempenho apenas desse trecho do código para as três versões, avaliando tempo de processamento do trecho e *speedup* ao se variar a quantidade de *threads* em 1, 2, 4 e 8 *threads*.

Atividade 3 - Seção crítica em Java

Considere o programa anexo em Java que implementa uma simulação de uma via de mão dupla, onde carros vêm em ambos os sentidos (direita e esquerda). Os carros vindos das

duas direções têm que passar por uma ponte, onde somente entra um carro por vez (tanto vindo da direita ou esquerda). Implemente um mecanismo de seção crítica que permita evitar colisões dos carros sobre a ponte de via única. O código implementa uma interface gráfica facilmente manipulável que permite inserir carros em ambas as direções, onde é possível identificar as colisões sobre a ponte.

Link para download do programa em Java:

https://drive.google.com/file/d/1gjP3E2F0ppLvgItm-PmW8xRe9ObWDrF2/view?usp=sharing