UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO

SISTEMAS OPERACIONAIS
PROF. JOÃO MARCOS BASTOS CAVALCANTI

Trabalho Prático I

*** Data de entrega: até 23 e 24/05/2019 ***

Este trabalho prático envolve programação concorrente. O objetivo do trabalho é familiarizar os alunos com a programação usando *threads,* os conceitos de comunicação entre *threads* por meio de compartilhamento de memória e o controle de concorrência usando semáforos e monitores. Considere no contexto deste trabalho apenas *threads.* Não devem ser implementados processos independentes.

1. Leitores × Escritores

Crie uma estrutura de dados (ex: um registro com vários campos ou um vetor), que será compartilhada por várias *threads* dos tipos leitores e escritores. As *threads* leitoras devem exibir os dados lidos e as threads escritoras devem atualizar e exibir os dados após a atualização.

O trabalho pode ter um "tema" como o problema do banco, portal do aluno, banco de dados de clientes de uma loja, registro de compras, etc.

Implemente três (3) versões:

1. Leitores e escritores sem controle de concorrência. Identifique os problemas causados pela falta de controle de concorrência.
2. Leitores e escritores com mesma "preferência". Pode ocorrer "leitura suja".
3. Escritores com preferência sobre leitores. Nesse caso, leitores somente têm acesso aos dados quando não há nenhum escritor ativo. Não ocorre "leitura suja".

O programa deve receber como entrada alguns parâmetros para as threads (ex: via teclado ou arquivo de entrada) com a quantidade de threads leitoras e escritoras e os valores que serão atualizados pelos escritores.

A saída do programa deve ilustrar o funcionamento do sistema, mostrando quando cada *thread* é criada, quando elas entram e saem da região crítica, quando elas são bloqueados, seu tipo (leitor ou escritor), o que fizeram e o momento em que a *thread* é finalizada.

2. Produtores × Consumidores - Problema do Barbeiro Dorminhoco

Em uma barbearia há um ou mais barbeiros, uma cadeira para cada barbeiro atender seus clientes e n cadeiras para espera. Quando não há clientes o barbeiro vai dormir. Quando chega um cliente, este precisa acordar um barbeiro para ser atendido. Se outros clientes chegarem enquanto todos os barbeiros estiverem ocupados, eles se sentarão nas cadeiras e aguardarão sua vez ou caso as cadeiras de espera estejam todas ocupadas os clientes vão embora.

Faça um programa que coordene as threads barbeiros e clientes. O programa deverá mostrar o "estado" de cada thread, por exemplo, "barbeiro atendendo", "barbeiro dormindo", "cliente i esperando" e outras informações como "salão lotado", etc.

OBSERVAÇÕES SOBRE A IMPLEMENTAÇÃO:

- A implementação deve obrigatoriamente incluir 2 (duas) versões para questão, uma usando semáforos e outra usando monitores. Exceção ao item 1 da primeira questão, que não tem controle de concorrência.
- Sugerimos a utilização da linguagem Java usando a classe Thread para a solução com monitores e C ou C++, por meio das bibliotecas "pthread" e "semaphore" para a solução com semáforos.
- Deve-se utilizar atrasos, por exemplo usando a função *sleep*, para deixar as *threads* com diferentes velocidades. Isso torna os resultados bastante variados e mais interessantes. Em geral as *threads* barbeiros / escritores são mais lentos que os clientes / leitores, mas deve ser possível testar qualquer configuração de atraso para qualquer tipo de *thread*.

OBSERVAÇÕES SOBRE A ENTREGA DO TRABALHO:

- O trabalho pode ser feito em equipe de no máximo 3 (três) alunos.
- O trabalho deverá ser apresentado ao professor, o qual fará algumas perguntas aos autores do trabalho.
- Os arquivos com código fonte dos programas também deverão ser entregues com identificação da equipe.
- Deve-se registrar a(s) fonte(s), seja URL, livro, slides, etc, de onde foram obtidos códigos fontes para desenvolvimento do trabalho, se for o caso.