Trabalho prático

Sincronização e Controle de Acesso

Objetivo

O objetivo deste trabalho é estimular o projeto, implementação e avaliação de soluções para problemas por meio de programação concorrente, em especial colocando em prática os conceitos e mecanismos de sincronização de processos/*threads*.

Problemas

Problema 1: O Banheiro Unissex

Um escritório contém um banheiro que pode ser utilizado tanto por homens quanto por mulheres, mas não por ambos ao mesmo tempo. Se um homem estiver no banheiro, outros homens podem entrar, porém eventuais mulheres que desejem utilizar o banheiro devem esperar ele ficar vazio. Se uma mulher estiver no banheiro, outras mulheres podem entrar, porém eventuais homens que desejem utilizar o banheiro devem esperar ele ficar vazio. Cada pessoa (homem ou mulher) pode passar um determinado tempo utilizando o banheiro, que possui uma capacidade limite de pessoas que podem utilizá-lo ao mesmo tempo.

Projete e implemente uma solução concorrente para o problema. O programa deve exibir a entrada e saída de uma pessoa (homem ou mulher) do banheiro bem como quantas pessoas (homens ou mulheres) estão no banheiro no momento. Por ser um espaço de tamanho relativamente diminuto, o banheiro possui uma capacidade limite de pessoas C (fornecida como entrada via linha de comando ou prefixada como um valor constante) que podem utiliza-lo ao mesmo tempo e o tempo que cada pessoa passa no banheiro é randômico e diferente a cada execução do programa.

Problema 2: Uma Lista Simplesmente Encadeada Concorrente

Considere uma lista simplesmente encadeada cujo acesso é compartilhado por três tipos de *threads*: o tipo *B* realiza operações de busca sobre a lista, o tipo *I* realiza operações de inserção de itens no final da lista e o tipo *R* realiza operações de remoção de itens a partir de qualquer posição da lista. *Threads* do tipo *B* meramente realizam operações de leitura sobre a lista e, portanto, podem ser executadas de forma simultânea com as outras. Por sua vez, as operações de inserção realizadas pelas *threads* do tipo *I* devem ser mutuamente exclusivas a fim de impedir que duas *threads* estejam inserindo itens no final da lista ao mesmo tempo, ainda que seja possível realizar essa operação de forma simultânea a uma operação de busca. Por fim, no máximo uma *thread* do tipo *R* pode acessar a lista por vez para realizar remoção de itens e essa operação deve ser mutuamente exclusiva com relação às demais (busca e inserção).

Projete e implemente uma solução concorrente para o problema que satisfaça esses três tipos de exclusão mútua entre as operações realizadas pelas *threads* dos tipos *B*, *I* e *R* sobre a lista simplesmente encadeada compartilhada. Por questões de simplicidade, considere que a lista armazena apenas valores do tipo inteiro. A cada (tentativa de) operação,o programa deverá exibir na saída padrão a operação que está sendo realizada.

Tarefas

A tarefa central a ser realizada neste trabalho consiste em projetar e implementar uma solução concorrente para cada um dos problemas anteriormente descritos utilizando conceitos e técnicas de programação concorrente, incluindo a criação, execução e sincronização de processos/*threads* independentes e concorrentes. A solução poderá ser implementada utilizando facilidades providas pelas linguagens de programação C/C++, Java **ou** Python, cada um utilizando **um mecanismo diferente** para sincronização de processos/*threads* concorrentes, ou seja, um determinado mecanismo utilizado na solução de um dos problemas **não poderá ser utilizado** na solução do outro.

O desenvolvimento da solução deve de antemão visar pela busca de desenvolvimento de *software* de qualidade, isto é, funcionando correta e eficientemente, exaustivamente testado, bem documentado e com tratamento adequado de eventuais exceções. Mais ainda, a implementação deverá garantir corretude do programa com relação a concorrência e aplicar de forma adequada os conceitos e mecanismos de sincronização, impedindo, portanto, a ocorrência de condições de *deadlock*, *livelock* ou *starvation* e realizando exclusão mútua de forma apropriada.

Além da implementação da solução, deverá ser elaborado um relatório escrito simples descrevendo, pelo menos:

- como a solução foi projetada;
- a lógica de sincronização utilizada, em termos dos mecanismos empregados e como ela é feita entre os fluxos de execução do programa;
- como é garantida a corretude da solução com relação a concorrência;

- eventuais dificuldades encontradas durante o desenvolvimento, e;
- instruções para compilação e execução do programa.

Autoria e política de colaboração

O trabalho poderá ser feito **individualmente ou em equipe composta por no máximo dois estudantes**, sendo que, neste último caso, é importante, dentro do possível, dividir as tarefas igualmente entre os integrantes da equipe. O trabalho em cooperação entre estudantes da turma é estimulado, sendo aceitável a discussão de ideias e estratégias. Contudo, tal interação não deve ser entendida como permissão para utilização de (parte de) código fonte de outras equipes, o que pode caracterizar situação de plágio. Trabalhos copiados em todo ou em parte de outras equipes ou da Internet serão sumariamente rejeitados e receberão nota zero.

A critério do professor, qualquer equipe pode ser eventualmente convocada para uma entrevista cujo objetivo é confirmar a autoria do trabalho desenvolvido e determinar a contribuição real de cada integrante. Durante a entrevista, cada membro da equipe deverá ser capaz de explicar, com desenvoltura, qualquer parte do trabalho, mesmo que esta tenha sido desenvolvida por outro membro da equipe. Portanto, é possível que ocorra, após a entrevista, redução da nota geral do trabalho ou ajustes nas notas individuais com vistas a refletir a verdadeira contribuição de cada membro da equipe.

Entrega

O trabalho deverá ser entregue até as **23h59 do dia 27 de maio de 2018** através da opção *Tarefas* da Turma Virtual do SIGAA. Deverá ser submetido um único arquivo compactado no formato `.zip` contendo todos os códigos fonte referentes à implementação das soluções para os problemas, disponibilizados sem erros e devidamente testados e documentados, além do relatório escrito, preferencialmente em formato PDF. Se for o caso, é possível fornecer, no campo *Comentários* do formulário eletrônico de submissão da tarefa, o endereço de um repositório remoto destinado ao controle de versões, porém esta opção não exclui a necessidade de submissão dos arquivos via SIGAA.

Avaliação

A avaliação deste trabalho será feita principalmente sobre os seguintes critérios: (i) utilização correta dos conceitos e mecanismos de sincronização de processos/*threads*; (ii) a corretude da execução dos programas implementados, tanto com relação a funcionalidades quanto a concorrência; (iii) a aplicação de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte, e; (iv) qualidade do relatório produzido. O trabalho possuirá nota máxima de 10,0 (dez) pontos e será contabilizada para a segunda unidade da disciplina.