UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
DIM0612 – PROGRAMAÇÃO CONCORRENTE | 2017.1

Trabalho prático

Sincronização em programas concorrentes

Objetivo

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

Problemas

Problema 1: Uma Montanha-Russa

Uma montanha-russa é uma atração popular dos parques de diversão e parques temáticos modernos, consistindo de uma estrutura de aço formando uma trilha composta por elevações seguidas de quedas e por vezes inversões. A maioria das montanhas-russa tem vários carros em que os passageiros sentam-se e são contidos (obviamente por questões de segurança), enquanto que algumas funcionam com um carro único e um número maior de passageiros.

Suponha uma montanha-russa contendo um único carro com capacidade para C passageiros. Os passageiros, em um total de n ($n > C$), frequentemente esperam para entrar no carro, que só pode ser liberado para a trilha quando está cheio. Após terminar um passeio, cada passageiro continua passeando pelo parque de diversões antes de retornar à montanha-russa para tentar novamente entrar no brinquedo em um outro passeio. Por questões de segurança, o carro realiza apenas P passeios por dia e, uma vez alcançado esse número, é desligado.

Projete e implemente programa que simule a operação da montanha-russa satisfazendo os seguintes requisitos:

- o carro permite a entrada de exatamente C passageiros;
- nenhum passageiro pode entrar no carro enquanto ele estiver em movimento sobre a trilha;
- nenhum passageiro pode pular do carro enquanto ele estiver em movimento sobre a trilha;
- nenhum passageiro poderá entrar novamente no carro sem que primeiro saia dele.

Durante a execução do programa, as seguintes ações devem ser realizadas para cada entidade:

- as ações realizadas por passageiros são o embarque (`board`) e o desembarque (`unboard`) do carro;
- o carro permite a entrada (`load`) e saída (`unload`) de passageiros;

- o movimento do carro é iniciado pela execução da operação `run`;
- os passageiros não podem entrar no carro até que a operação `load` seja executada pelo carro;
- o carro não pode iniciar seu movimento até que todos os C passageiros tenham entrado;
- os passageiros não podem sair do carro até que a operação `unload` seja executada pelo carro.
- o tempo em que cada passageiro passeia pelo parque antes de ir novamente à montanha-russa é randômico e diferente a cada execução do programa.

Para fins de legibilidade, o programa deverá exibir na saída padrão a execução de cada uma das operações realizadas pelos passageiros e pelo carro. O número total de passageiros n, a capacidade do carro C e o número máximo de passeios P que podem ser realizados por dia deverão ser fornecidos como entrada ao programa como argumentos de linha de comando. Caso $C < n$, o programa deverá emitir uma mensagem de erro para o usuário e ser imediatamente encerrado.

Extra: Desenvolva uma segunda versão do programa para fazer com que a montanha-russa comporte múltiplos carros. Essa segunda versão não é uma simples generalização da anterior em que tem-se agora m carros de igual capacidade C (ao invés de apenas um), mas também satisfaz o seguinte conjunto de restrições adicionais:

- o embarque de passageiros só pode ser feito em um carro por vez;
- os múltiplos carros podem estar simultaneamente na trilha;
- uma vez que um carro não pode ultrapassar outro na trilha, o desembarque de passageiros tem de ser feita na mesma ordem em que eles embarcaram;
- os passageiros de um carro só podem desembarcar dele quando todos os passageiros do carro que está a sua frente já tenham saído.

Problema 2: 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 tempo utilizando o banheiro.

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.

Tarefas

A tarefa central a ser realizada neste trabalho é conceber e implementar uma solução para cada um dos problemas anteriormente descritos utilizando conceitos e técnicas de programação concorrente,

incluindo a criação, execução e sincronização de fluxos de execução (processos/*threads*) independentes e concorrentes. A solução poderá ser implementada utilizando facilidades providas pelas linguagens de programação C/C++, Java **ou** Python e **pelo menos dois** dos mecanismos de sincronização de processos/*threads* concorrentes vistos na disciplina.

A implementação deverá garantir corretude do programa com relação a concorrência e aplicar de forma adequada os conceitos e mecanismos de sincronização. O desenvolvimento da solução deve de antemão visar pela busca de desenvolvimento de *software* de qualidade, isto é, funcionando correta e eficientemente, exaustivamente testado, obrigatoriamente bem documentado e com tratamento adequado de eventuais exceções.

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, ou seja, como a implementação feita garante exclusão mútua apropriada e ausência de potenciais condições de *deadlock* e/ou *starvation*;
- uma análise comparativa entre os diferentes mecanismos de sincronização utilizados, em particular com relação a esforço de desenvolvimento e complexidade;
- 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 31 de maio de 2017, prazo que não será estendido**. Exclusivamente através da opção *Tarefas* da Turma Virtual do SIGAA, deverá ser submetido um único arquivo compactado 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 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* vistos na disciplina; (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.