Trabalho Prático 1

Algoritmo do Banqueiro

Nesse trabalho prático, você deverá escrever um programa multithreaded que implemente o algoritmo do banqueiro discutido na Seção 7.5.3 do livro[1]. Vários clientes solicitam e liberam recursos do banco. O banqueiro atenderá uma solicitação somente se ela deixar o sistema em um estado seguro. Uma solicitação que deixe o sistema em um estado inseguro será negada.

Essa tarefa de programação combina três tópicos diferentes: (1) criar múltiplos threads, (2) prevenir condições de corrida e (3) evitar deadlocks.

O Banqueiro

O banqueiro considerará solicitações de n clientes por m tipos de recursos, como descrito na Seção 7.5.3. O banqueiro controlará os recursos usando as estruturas de dados a seguir:

```
/* estes podem ser quaisquer valores >= 0 */
#define NUMBER_OF_CUSTOMERS 5
#define NUMBER_OF_RESOURCES 3
/* o montante disponível de cada recurso */
int available[NUMBER_OF_RESOURCES];
/* a demanda máxima de cada cliente */
int maximum[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
/* o montante correntemente alocado a cada cliente */
int allocation[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
/* a necessidade remanescente de cada cliente */
int need[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
```

Os Clientes

Crie *n* threads de clientes que solicitem e liberem recursos do banco. Os clientes estarão em um loop contínuo, solicitando e depois liberando números aleatórios de recursos. As solicitações dos clientes por recursos serão limitadas por seus respectivos valores no array need. O banqueiro atenderá uma solicitação se ela satisfizer ao algoritmo de segurança descrito na Seção 7.5.3.1 do livro[1]. Se uma solicitação não deixa o sistema em um estado seguro, o banqueiro a negará. Os protótipos das funções para a solicitação e liberação de recursos são os seguintes:

```
int request_resources(int customer_num, int request[]);
int release_resources(int customer_num, int release[]);
```

---

[1] *SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. Fundamentos de sistemas operacionais. 9. ed. Rio de Janeiro, RJ: LTC, c2015. E-book. ISBN 978-85-216-3001-2. (Livro Eletrônico).*

Essas duas funções devem retornar 0, se bem-sucedidas, e -1, se não houver êxito. Múltiplos threads (clientes) acessarão concorrentemente dados compartilhados por meio dessas duas funções. Portanto, o acesso deve ser controlado por meio de locks mutex para prevenir condições de corrida. As APIs Pthreads e Windows fornecem locks mutex. O uso de locks mutex do Pthreads é abordado na Seção 5.9.4 do livro[2]; os locks mutex dos sistemas Windows são descritos no projeto intitulado “Problema do Produtor-Consumidor” no fim do Capítulo 5 do livro[2].

Implementação Você deve invocar seu programa passando o número de recursos de cada tipo na linha de comando. Por exemplo, se houver três tipos de recursos, com dez instâncias do primeiro tipo, cinco do segundo tipo e sete do terceiro tipo, você invocaria seu programa assim:

$ ./a.out 10 5 7

O array available seria inicializado com esses valores. Você pode inicializar o array maximum (que contém a demanda máxima de cada cliente) usando qualquer método que achar conveniente.

Entrega

Para este trabalho prático deverá ser entregue um relatório no canvas no formato PDF na data especificada na tarefa. Neste relatório deverá conter um link para um repositório público no Github com o código acompanhado de um arquivo readme.md com as instruções de como realizar a compilação e a execução. A escolha da linguagem de programação é de responsabilidade dos alunos. O trabalho poderá ser feito individualmente ou em dupla. Em caso de dupla, ambos os alunos deverão submeter o relatório no canvas.

Avaliação

O trabalho prático será avaliado da seguinte forma: **5 pontos** do relatório e **5 pontos** da implementação totalizando **10 pontos**. Somente a entrega do relatório ou somente do projeto de implementação não será aceito, sendo assim a nota será 0. O relatório deverá conter uma **introdução** sobre o tema, seguido de três seções: **desenvolvimento**, **resultados** e **conclusão**. A implementação será avaliada conforme a corretude do código e a organização do repositório.

---

[2] *SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. Fundamentos de sistemas operacionais. 9. ed. Rio de Janeiro, RJ: LTC, c2015. E-book. ISBN 978-85-216-3001-2. (Livro Eletrônico).*