Controle e Simulação de Rodadas Grátis em um Bar

O presente trabalho tem por objetivo explorar o uso de threads e seus mecanismos de sincronização e controle de acesso a regiões críticas.

Descrição do problema

Um bar resolveu liberar um número específico de rodadas grátis para seus ***n*** clientes presentes no estabelecimento. Esse bar possui ***x*** garçons. Cada garçom consegue atender a um número limitado (***C***) de clientes por vez. Como essas rodadas são liberadas, cada garçom somente vai para a copa para buscar o pedido quando todos os ***C*** clientes que ele pode atender tiverem feito o pedido ou não houverem mais clientes a serem atendidos. Após ter seu pedido atendido, um cliente pode fazer um novo pedido após consumir sua bebida (o que leva um tempo aleatório) e a definição de uma nova rodada liberada. Uma nova rodada somente pode ocorrer quando foram atendidos todos os clientes que fizeram pedidos. Por definição, nem todos os clientes precisam pedir uma bebida a cada rodada.

Construção da Solução

Implemente uma solução que permita a passagem por parâmetro para o programa: (i) o número de clientes presentes no estabelecimento; (ii) o número de garçons que estão trabalhando; (iii) a capacidade de atendimento dos garçons; e (iv) o número de rodadas que serão liberadas no bar. Cada garçom e cada cliente devem ser representados por threads, estruturalmente definidos como os pseudo-códigos que seguem:

```
thread cliente {
   while (!fechouBar){
      fazPedido();
      esperaPedido();
      recebePedido();
      consomePedido(); //tempo variável
   }
}
```

```
thread garçom {
   while(existemClientesNoBar {
      recebeMaximoPedidos();
      registraPedidos();
      entregaPedidos();
       rodada++; //serve como parâmetro para
                 // fechar o bar
   }
}
```

Este trabalho deve ser implementado em C++, usando a biblioteca POSIX PThread, no Linux, utilizando semáforos e threads.

- O principal desafio do trabalho é o uso de semáforos para controlar concorrência e a sincronização entre as threads.

A ordem de chegada dos pedidos dos clientes na fila de pedidos de cada garçom deve ser respeitada.

- A sua solução não deve permitir que clientes furem essa fila.

O garçom só pode ir para a copa quando tiver recebido seus ***C*** pedidos. O programa deve mostrar a evolução, portanto planeje bem o que será apresentado. Deve ficar claro o que está acontecendo no bar a cada rodada. Os pedidos dos clientes, os atendimentos pelos garçons, os deslocamentos para o pedido, a garantia de ordem de atendimento, etc.

Formato de Entrega e Avaliação

O trabalho individual deverá ser **apresentado** em sala de aula (5 minutos por aluno) no dia especificado. Todos os arquivos contendo o código do trabalho, bem como Makefile e um **relatório** apresentando **sucintamente** a solução, deverão ser submetidos pelo moodle. Não serão aceitos trabalhos entregues fora do prazo. Trabalhos que não compilam ou que não executam receberão nota ZERO, bem como trabalhos que sejam considerados como plágio. Erros de português terão 0.2 de desconto para cada um (ortografia e concordância).

Os itens para avaliação são: (i) funcionamento do programa; (ii) execução das threads sem ocorrência de deadlocks e/ou outros problemas de sincronização; (iii) saída do programa (de modo a permitir a avaliação de seu funcionamento); (iv) clareza do código (utilização de comentários e nomes de variáveis adequadas); (v) interface e usabilidade; (vi) apresentação do trabalho; (vii) qualidade do relatório; (viii) compilação sem warnings; (iv) sem vazamento de memória e (v) modelagem do software desenvolvido com diagramas UML.

Referências

- Disciplina de Sistemas Operacionais do curso de Ciência da Computação da PUC-RS. Professor Edson Ifarraguire Moreno.
- http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
- https://computing.llnl.gov/tutorials/pthreads/
- http://www.bogotobogo.com/cplusplus/multithreading_pthread.php
- Thread support library - http://en.cppreference.com/w/cpp/thread