Universidade Federal de Santa Catarina

Bar das threads
Professor:Gian Ricardo Berkenbrock
Aluno:Bruno Mamoru Kato Kawakami
Matrícula: 15102609

Introdução

O objetivo do trabalho em questão foi a implementação utilizando threads e ferramentas auxiliares ao multithreading (como mutexes e variáveis de condição), de um bar na linguagem de programação C++ .

Há neste uma determinada quantidade de clientes que deverão ser servidos, assim como uma quantidade de garçons, capacidade máxima de atendimento de cada garçom e número total de rodadas que serão servidas. Adicionalmente foi utilizada uma nova variável que determina a taxa de pedidos de um cliente, se este irá pedir ou não na rodada.

Optou-se pela programação orientada a objetos devido a uma maior clareza e facilidade de entendimento do código. Criou-se classes para representar cada um dos seguintes componentes, bar, garçons, clientes e pedidos, onde cada uma possui suas determinadas características e funções.

1 - Classes

1.1 - Classe Bar

A classe bar é a responsável pelo controle das threads garçons e clientes que estarão executando durante todo o programa, checando a situação de suas variáveis. Esta possui vetores de ponteiros para garçons e para clientes, além de outras variáveis como um vetor de pedidos e um de produtos, capacidade de garçons, quantidade de clientes, mutexes, variável de condição, que serão utilizadas para realizar a sincronização das threads.

Devido ao método de implementação adotado a maioria de seus atributos e de seus métodos são estáticos, possibilitando o uso destes sem a necessidade de instanciar um objeto.

Além disso, houve a necessidade de alocar memória para os objetos que usariam as threads, para isso utilizou-se de um recurso do C ++ 14, que sao os

ponteiros inteligentes, os utilizados no caso foram os “std::unique_ptr” e para sua instanciação foi utilizado o “std::make_unique”.

1.2 - Classe Cliente

A classe cliente é a responsável por simular o comportamento do cliente. Esta possui como atributo uma thread que será inicializada quando o bar executar o método “run” da classe, que por sua vez será executada somente quando o bar for iniciado ou seja quando o método “inicio” for chamado.

Após este momento a thread irá esperar com que todas as outras threads estejam prontas para só então começar a executar a sua rotina, que consiste em esperar a rodada começar, método “começa” que age como uma barreira esperando que uma nova rodada comece, realizar um pedido, esperar o pedido ser entregue e então consumir o pedido.

Todos os métodos citados acima utilizam de mutex e variáveis de condição para que haja uma sincronização entre as threads.

1.3 - Classe Garçom

A classe garçom é a responsável por simular o comportamento do garçom. Esta, assim como a classe, cliente possui como atributo uma thread que será inicializada quando o bar executar o método “run” da classe.

Similarmente à thread do cliente a do garçom irá esperar com que todas as outras estejam prontas para só então começar a executar sua rotina, que consiste no método “começa”, este espera com que todos os clientes que possuem o desejo de pedir façam seus pedidos (adicionem seus pedidos ao buffer de pedidos do bar) , seguido pelo método “pegaMaxPedidos”, que irá coletar todos os pedidos possíveis (até possuir sua capacidade máxima, ou não haver mais pedidos) do buffer de pedidos do bar e adicionar em seu próprio buffer de pedidos.

Feito isso o próximo método a ser chamado será o “produzPedidos” que irá fazer com que o garçom produza os pedidos coletados (retire os pedidos do buffer de pedidos do garçom, coletados anteriormente, e adiciona no buffer de produtos do bar).

1.4 - Classe Pedido

A classe Pedido será a responsável do armazenar a identidade do cliente que realizou determinado pedido, e também a do garçom que atendeu o pedido.

Esta possui métodos para retornar a identidade dos clientes, identidade dos garçons e o numero do pedido realizado pelo cliente.

2 - Funcionamento

Inicialmente na “main” o bar será instanciado, sendo neste momento passados a ele a quantidade de clientes, a quantidade de garçons, a capacidade de atendimento de cada garçom e o número máximo de rodadas do bar.

Feito isso será chamada o método “inicio” que irá adicionar as threads clientes e as threads garçons e fará com que elas executem sua rotina.

A thread cliente irá executar sua rotina, que como citado anteriormente, consiste em esperar com que uma nova rodada seja liberada, método “começa” que age como uma barreira ,e somente libera os clientes para realizar os pedidos se todos os garçons terminaram de atender os pedidos solicitados e após todos os clientes conseguirem pegar os seus respectivos pedidos. Para tal utiliza-se um método de classe do bar que checa se uma nova rodada pode iniciar, além de empregar um mutex e uma variável de condição, em que o primeiro será utilizado para conseguir acesso à região crítica, e o segundo para fazê-lo ser bloqueado, caso as condições não sejam atendidas, e posteriormente notificar a liberação do mutex para as outras threads.

Além disso o método “novaRodada” do bar utilizada pelo cliente sorteará novas intenções de pedidos para os clientes, se estes irão pedir ou não nesta rodada, através do método “sorteiaPedido” do cliente, que dependendo de seu “desejo” (uma enum class que possui os estados, “encher a cara”, “passar o tempo” e “dia de maldade”), sorteado aleatoriamente no construtor do bar, terão maior ou menor chance de realizar um pedido na rodada, estando a chance de pedir na ordem crescente. Em adição, aqueles clientes que estão no estado “consumindo” não poderão pedir caso a rodada comece e eles estiverem ainda neste estado.

Após ser liberado para realizar o pedido, o mesmo tenta novamente obter o mutex para entrar na região crítica e adicionar o seu pedido no buffer do bar.

Feito isso o cliente irá esperar até que a thread garçom produza seu pedido, ou seja este ficará bloqueado devido ao uso da variável de condição, até que encontre no primeiro lugar do buffer de produtos do bar o seu pedido, garantindo deste modo que, o primeiro a pedir será sempre o primeiro a receber o pedido.

Enquanto o cliente espera o seu pedido, o garçom, que ficará aguardando, bloqueado, até que todos os clientes realizem o pedido, se assim desejarem, caso passe pela barreira, pois todos os clientes já realizaram seu pedido, recolhe todos os pedidos que consegue do buffer de pedidos (até que atinja sua capacidade máximo, ou até que não haja mais pedidos no buffer do bar) , produz e entrega.

Quando todos os pedidos feitos foram atendidos, os garçons ficarão esperando novamente na barreira até que os clientes consumam o produto que lhes foi entregue e uma nova rodada comece.

Este processo irá continuar até que todos os clientes tenham consumido o número de rodadas que o bar fornece.

O ciclo de atendimento somente será encerrado quando todos os clientes tiverem saído bar, somente então os garçons deixarão suas rotinas e o bar poderá caminhar para o fechamento.

No final do programa, quando o destrutor do bar é chamado, toda a memória alocada utilizando o ponteiro inteligente é automaticamente desalocado, após a instanciação do bar ser destruída.