TRABALHO FINAL DA DISCIPLINA DE PROGRAMAÇÃO CONCORRENTE

Guilherme Silva Souza - 19/0014059

23 de outubro de 2021

1 Introdução

O que é programação concorrente? Descrito de forma simples, é quando você está fazendo mais de uma coisa ao mesmo tempo. Ela não deve ser confundido com paralelismo, concorrência é quando várias sequências de operações são executadas em períodos de tempo sobrepostos. No reino da programação, a concorrência é um assunto bastante complexo. Lidar com construções como *threads* e bloqueios e evitar problemas como condições de corrida e *deadlocks* pode ser bastante complicado, dificultando a escrita de programas concorrentes. Por meio da concorrência, os programas podem ser projetados como processos independentes trabalhando juntos em uma composição específica. Tal estrutura pode ou não ser paralela; entretanto, alcançar tal estrutura em seu programa oferece inúmeras vantagens. Isso não significa necessariamente que vários processos estarão em execução ao mesmo tempo - mesmo que os resultados façam parecer que sim.

Algumas vantagens da programação concorrente incluem o aumento de desempenho, pois aumenta-se a quantidade de tarefas sendo executadas em determinado período de tempo, e a possibilidade de uma melhor modelagem de programas, pois determinados problemas computacionais são simultâneos por natureza.

O proposto trabalho tende-se a se basear em um problema do mundo real trazido para o contexto da programação concorrente. É desenvolvido um pequeno sistema onde simula o funcionamento de um supermercado. Onde nele há limite de entrada de clientes, estoques com produtos finitos e funcionários que possam repor o estoque. Nesse problema enfrentamos desafios como *deadlocks* que se refere a uma situação em que ocorre um impasse, e dois ou mais processos ficam impedidos de continuar suas execuções - ou seja, ficam bloqueados, esperando uns pelos outros. E condições de corrida que é uma situação onde os eventos que ocorrem podem influenciar na execução de processos, especialmente quando existem vários fluxos de execução "simultâneos", e também soluções para esse problema. Por exemplo vários clientes do supermercado comprando o mesmo produto em que esse produto tem um estoque limitado.

2 Conceitos utilizados nesse trabalho

2.1 Threads

Thread é um pequeno programa que trabalha como um subsistema, sendo uma forma de um processo se auto dividir em duas ou mais tarefas. É o termo em inglês para Linha ou Encadeamento de Execução. Essas tarefas múltiplas podem ser executadas simultaneamente para rodar mais rápido do que um programa em um único bloco ou praticamente juntas, mas que são tão rápidas que parecem estar trabalhando em conjunto ao mesmo tempo.

2.2 Locks

É uma ferramenta que possibilita a sincronização entre processos em que processos/*threads* devem ser programados de modo que seus efeitos sobre os dados compartilhados sejam equivalentes serialmente.

2.3 Variáveis de Condição

Variáveis de condição servem para controlar o acesso de uma *thread* sobre determinada região do programa. Essa *thread* pode ser posta em *sleep*, na qual ela é bloqueada e espera o sinal de *wakeup*.

2.4 Semáforos

Os semáforos são uma primitiva de concorrência muito básica que suportam o acesso concorrente a zonas de memória partilhada, ainda que alguns problemas de sincronização possam surgir, tais como *deadlocks* e *starvation*. Os semáforos tem funcionamento muito parecidos com os locks, mas com o número de permissões maior. No caso de um semáforo de uma permissão, seu funcionamento será igual ao de um *lock*.

3 Formalizacão do Problema Proposto

A premissa do projeto é simular um supermercado. Supermercado é um grande comércio tradicional de alimentos, com um sistema de autosserviço que oferece uma grande variedade de alimentos e produtos domésticos, organizados em corredores. É maior em tamanho e tem uma vasta seleção de uma mercearia tradicional, mas é menor e mais limitado na gama de mercadorias do que um hipermercado.

Diariamente os supermercados recebem um número grande de clientes querendo fazer suas compras, cada cliente busca um ou mais produtos específicos, e podem ter vários clientes procurando o mesmo produto. Nesse contexto entra a condição de corrida que são situações onde dois ou mais processos (clientes) estão acessando dados compartilhados (produtos), e o resultado final do processamento depende de quem executa e quando executa.

Claro que a variedades de produtos e o número de itens em supermercado real é bem alto, mas nesse projeto só foi colocado 4 produtos e um estoque limitado para que ocorra a concorrência entre os clientes.

4 Solução do Problema Proposto

A solução do problema proposto foi feita na linguagem C, utilizando a biblioteca *pthreads* que foi vista em sala de aula durante todo o semestre. É utilizada 5 funções para dar suporte ao fluxo do supermercado. Temos dois tipos de *threads* para essa simulação: um para os clientes e outra para os funcionários do supermercado.

O supermercado tem um estoque de quatro produtos: arroz, feijão, macarrão e ovos. Cada produto inicia seu estoque com N itens. Quando algum produto zerar seu estoque, seus itens tem que ser reposto por um funcionário do supermercado. Esse estoque é representado na forma de um *array* de inteiros de 4 posições, onde cada posição representa um produto, arroz, feijão, macarrão e ovos, respectivamente. E cada posição guarda a quantidade de itens no estoque.

Na entrada do supermercado tem um limite de N clientes que vão poder fazer suas compras. Cada cliente entra no supermercado e escolhe um produto e quantos itens daquele produto ele vai comprar. Logo depois o cliente se direciona ao caixa para finalizar sua compra, na hora que ele finaliza a compra o cliente sai do supermercado e é contabilizada uma operação feita naquele dia para o supermercado. O supermercado fecha as portas quando n ou mais operações forem realizadas.

Para controlar o número de clientes que vão poder entrar no supermercado foi utilizado um semáforo com n permissões. Assim quando o número de permissões chega a 0, nenhum cliente vai poder entrar no supermercado.

Dentro do supermercado cada cliente irá comprar um produto e n itens daquele produto, sempre respeitando um número máximo de itens que o cliente poderá comprar. Cada produto tem um *lock* associado a ele, desse modo na hora da compra só um cliente por vez poderá retirá-lo do estoque.

Quando um produto do estoque é zerado, uma *flag* é acionada para sinalizar a falta daquele produto, quando a *flag* tem o valor -1 significa que nenhum produto do estoque está em falta, quando o cliente identifica um produto fora de estoque, ele seta o valor da *flag* para o índice do produto no *array*. É utilizada uma variável de condição que tem a função de acordar o funcionário para que ele possa repor aquele produto no estoque. Assim que ele identifica o produto, ele repõe o mesmo no estoque e voltar a dormir.

Quando o cliente finaliza a compra, ele sai do supermercado e é adicionado 1 ao número de operações feitas pelo supermercado. Quando o número de operações passa de um certo valor n, o supermercado fecha e exibe algumas estatísticas sobre os produtos vendidos.

5 Conclusão

A principal vantagem do uso da programação concorrente é o aumento do desempenho dos programas, pois é possível aumentar a quantidade de tarefas executadas em um determinado período de tempo.

Esta abordagem é muito utilizada profissionalmente. Por exemplo, em uma grande empresa, pode-se calcular o faturamento mensal da empresa executando o cálculo do faturamento de forma concorrente, processando várias filiais simultaneamente. Neste caso, a programação concorrente permite ganho de tempo pois processamos todas as filiais ao mesmo tempo, ao invés de processar o faturamento de uma filial de cada vez.

Outra vantagem é que o projeto dos programas se torna mais simples com relação à concorrência entre os programas pelos recursos compartilhados.

6 Referências

https://www.techtudo.com.br/noticias/2019/01/o-que-sao-threads-e-para-que-servem-em-um-processador.ghtml

https://www.educative.io/edpresso/what-is-concurrent-programming

https://pt.wikipedia.org/wiki/Condi%C3%A7%C3%A3o_de_corrida

revista-programar.info/artigos/threads-semaforos-e-deadlocks-o-jantar-dos-filosofos/