UNIVERSIDADE FEDERAL DE SANTA CATARINA

BLU3020-08754 (20182) - SISTEMAS COMPUTACIONAIS PARA CONTROLE E AUTOMAÇÃO

LUCAS ENDO PRESTES

NÍCOLAS WILSON SOUZA

TRABALHO PRÁTICO 1

BLUMENAU, 26 DE SETEMBRO DE 2018

Sumário

1. INTRODUÇÃO

Este trabalho consiste de três estudos teóricos de caso e uma implementação prática, será iniciado este trabalho com estudo de problemas clássicos de sincronização entre processos, com abordagem nos seguintes temas:

- Jantar dos filósofos
- Jantar do Canibais
- Barbeiro Sonolento

Será feito uma abordagem em cada um dos temas em forma de resumo do problema, porém somente será implementado o problema 'Jantar dos canibais', ao final da implementação, será feito uma avaliação crítica, com intuito de indicar limitações da implementação e pontos para futuras melhorias, como limitação plausível, a implementação deverá ser livre de *DeadLocks* e a implementação deverá ter disponibilidade para alterar parâmetros relevantes, tal como número de *threads*, *etc.* No final da implementação, nós avaliaremos o resultado, com o intuito de apontar limitações da implementação e pontos de melhoria

2. DESENVOLVIMENTO

a. INTRODUÇÃO AOS PROBLEMAS GERAIS

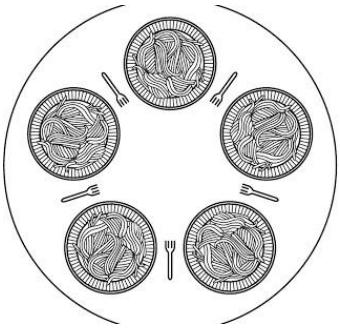
O trabalho dedicado a disciplina de Sistemas computacionais para Controle e Automação consiste em demonstrar problemas simples de serem interpretados e algumas vezes difíceis de serem consertados de modo eficaz e eficiente para todas as possibilidades de otimização, no mundo da programação mais atual, que começa a partir dos processadores com multi núcleo, surgindo problemas como a concorrência e condição de corrida. A programação concorrente, é uma parte da programação que tem como foco a execução, interação e análise de multitarefas. Existem vários problemas que servem para entender a questão de threads e sincronização de processos.

O presente trabalho tratará de três problemas comuns na iniciação de estudos na área de sistemas computacionais. Sendo que somente um desses problemas será resolvido na linguagem C que será compilado em GCC.

b. JANTAR DOS FILÓSOFOS

Cinco filósofos em torno de uma mesa de jantar redonda. um garfo é colocado entre cada filósofo juntamente com um prato à sua frente com espaguete. O espaguete está muito liso e o filósofo necessita de dois garfos para poder comer. Os dois garfos devem ser aqueles logo a sua esquerda e a sua direita. O filósofo deve alternadamente comer e pensar. O filósofo tenta pegar os garfos à sua direita e à sua esquerda, sendo um de cada vez, não importando a ordem de escolha. Após comer, o filósofo deve liberar o garfo que utilizou. Um filósofo pode segurar o garfo da sua direita ou o da sua esquerda assim que estiverem disponíveis, mas, só pode começar a comer quando ambos estiverem sob sua posse. Se um filósofo pegar um dos garfos e o outro a sua esquerda ou direita não estiver disponível, o mesmo deve soltar o garfo pego para que outro filósofo possa pegar.

Figura 1 - Representação Jantar dos Filósofos



Fonte: papeldiario.blogspot.com (2018)

A falta de garfos para os filósofos está relacionada a parte de recursos compartilhados em uma programação de computador. Quando um programa está interessado em um recurso que está sendo utilizado por outro programa, o mesmo aguarda até que seja terminado. Quando se tem mais de um programa envolvido na parte de no travamento de recursos, um *deadlock* pode ocorrer.

c. BARBEIRO SONOLENTO

No campo de ciências da computação, o problema do barbeiro dorminhoco é um problema conhecido de comunicação entre threads e sincronização entre elas. Este problema tem que programar o barbeiro e os clientes sem cair em condições de corrida. Essas condições são situações em que duas ou mais threads (ou processos) estão trabalhando juntas e podem compartilhar algum armazenamento. Sendo assim havendo um problema entre os threads, O problema abaixo pode dar um exemplo do que pode ocorrer.

Barbeiro dorminhoco: Na barbearia tem um barbeiro, uma cadeira de barbeiro e x cadeiras para os clientes esperarem a sua vez. Quando não há clientes, o barbeiro se senta na cadeira de barbeiro e dorme. Quando chega um cliente, ele precisa acordar o barbeiro. Se outros clientes chegarem enquanto o barbeiro estiver cortando o cabelo de um cliente, eles se sentarão (se houver cadeiras vazias) ou sairão da barbearia (se todas as cadeiras estiverem ocupadas). O problema é abstrato e deve manter o barbeiro ocupado enquanto há clientes, e descansando quando não há nenhum realizando assim de uma maneira ordenada.

Figura 2 – Representação Barbeiro Sonolento

Fonte: groups.ist.utl.pt (2018)

d. JANTAR DOS CANIBAIS

O problema do jantar dos canibais surge com uma condição de corrida entre duas diferentes threads, suponha que um grupo de N canibais come jantares a partir de uma grande travessa que comporta M porções. Quando alguém quer comer, ele(ela) se serve da travessa, a menos que ela esteja vazia (Onde N threads seriam N canibais se servindo). Se a travessa está vazia, um dos canibais acorda o cozinheiro e espera até que o cozinheiro coloque mais M porções na travessa (A chamada do cozinheiro ira desencadear uma segunda thread, sendo que esta, faz o processo de encher com um valor X a travessa dos canibais, enquanto o cozinheiro enche a travessa os canibais não podem estar comendo). O desenvolvimento do código para as ações dos processos canibais e do processo cozinheiro dará utilizando semáforos para sincronização. A solução com semáforos deverá evitar *deadlock também* e deve acordar o cozinheiro apenas quando a travessa estiver vazia, e deixar os canibais comerem somente após o cozinheiro encher a travessa com X porções.

3. CONSIDERAÇÕES FINAIS

Neste presente trabalho, nosso grupo teve dificuldade na implementação do trabalho em linguagem C. O problema que escolhemos para fazer a solução foi o dos Canibais que pode ser visto na página anterior (item d). Este problema se relaciona a parte de comunicação entre threads.

Tivemos dificuldades em trabalhar com a aplicação de *multithreads*, apesar do entendimento em sala de aula, foi um pouco complicado abordar na solução do problema. Pois elas precisavam trabalhar concorrentemente.

Outro problema que teve foi de resolver a parte de encontrar os erros na programação. A primeira vez que executamos haviam muitos erros que apareceram na compilação, após muitas tentativas e análise do programa em C, tivemos que recorrer para a ajuda do professor para resolver um problema específico, o qual nos deu uma ideia do que havia ocorrido e como resolver.

A parte do *While* foi um problema que também apareceu neste trabalho onde falhamos em ter deixado uma parte da programação fora do *while*, assim não executando corretamente o programa.

Ao fim do programa teve-se que acrescentar uma nova parte dentro da função *main*, a *pthread_join*. A mesma tem o objetivo de impedir que a função *main* finalize o programa, porque se não for colocado está thread, o *main* e também as outras threads dentro desta função são finalizadas. Outra dificuldade foi a questão de encontrar uma forma de colocar no prgramação o *fflush (inout)* para facilitar e nos ajudar.

Com todas as dificuldades, conseguimos resolver os problemas de programação e encontrar uma solução para o problema dos canibais. Foram resolvidos juntamente com a leitura de livros e colaboração do professor da disciplina. Sendo assim, um trabalho de grande aprendizado e grande valia para as partes envolvidas.