Sincronização de processos com the big bang theory

***Abstract*—Este trabalho descreve o projeto final da disciplina de Sistemas operacionais do curso de Ciência da Computação. Procurando avaliar o entendimento sobre sincronização de processos e tendo como base algoritmos clássicos como o Algoritmo do barbeiro e Jantar dos filósofos, foi proposto um problema de sincronização de processos no uso de um aparelho para simular o acesso à CPU por processos concorrentes. Foi avaliada a criatividade e abstração dos alunos para utilizar ferramentas como threads, semáforos e mutexes.**

***Index Terms*—Sincronização, Sistemas Operacionais, Educação.**

I. Introduction

SINCRONIZAÇÃO de processos é uma parte crucial do funcionamento de todos os sistemas operacionais modernos. Neste sentido, o presente projeto se propôs em representar uma interação entre personagens populares para simular a sincronização de processos acessando áreas críticas e variáveis compartilhadas em um programa.

A capacidade de abstração é muito importante para diversas áreas, inclusive na computação. Jeff Krammer[1] defende em seu trabalho que a diferença entre engenheiros e cientístas da computação que conseguem produzir soluções elegantes e limpas para problemas está nos seus níveis de abstração. Este projeto final da disciplina de Sistemas Operacionais procura desenvolver a capacidade de abstrair problemas computacionais em situações mundanas.

22 Julho, 2019

A. Descrição do problema

O personagens da famosa sitcom The Big Bang Theory comparam um microondas juntos e seu uso vai seguir certas diretrizes:

1) Se o microondas estiver livre, chegar primeiro usar.
2) Se mais estiver ocupado, a pessoa espera.
3) Se tiver mais alguém esperando, vale a seguinte precedência.
   - Sheldon pode usar antes do Howard;
   - Howard pode usar antes do Leonard;
   - Leonard pode usar antes do Sheldon.
4) quando alguém termina de usar o forno, deve liberá-lo para a
5) próxima pessoa de maior prioridade, exceto em dois casos:
   - para evitar inanição (discutido a seguir);
   - quando houver deadlock (um ciclo de prioridades).

Cada uma deles possui uma namorada que tem a mesma prioridade do parceiro. Caso haja Deadlocks, outro personagem chamado Raj deve tirar algúem da fila, este deve checar a situação a cada 5 segundos para saber se precisa interferir.

II. Implementação

A implementação utiliza threads, semáforos numéricos, mutexes e structs. Partes importantes do código:

A. Structs

- personagem: possui como parâmetros o nome as pessoas para quem cede a vez.
- personagens: possui a fila de no máximo 6 posições e o tamanho a ser acessado constantemente.
- disp: agrega informações sobre a disponibilidade do personagem.

B. Threads e funções

- Thread r: ativa a função ”raj()” que verifica a cada 5 segundos se está acontecendo algum problema na fila de espera.
- Thread m: rege a função ”microondas()” que espera durante 1 segundo representando o uso do aparelho e manda sinal para retirar a pessoa da fila.
- Thread v: ativa a função ”verdisp()” que vai procurar saber se o personagem está disponível para retornar à fila de espera para utilizar o microondas.
- addpersonagem(): escolhe aleatoriamente alguém disponível para colocar na fila.
- sortdecrescente(): organiza a fila seguindo as ordem de prioridade estabelecida pelo problema.
- ircomer2(personagem): representa o tempo que um personagem que acabou de utilizar o microondas fica fora da fila.

C. Inicialização - Função main

Inicialização de semáforos e threads, nó de repetição adicionando personagens indefinidamente.

III. Problema

Ao adicionar um novo personagem à fila, a fila é ordenada de acordo com a prioridade de cada pessoa, em casos específicos em que pelo menos 3(três) personagens como Sheldon, Howard e Leonard estiverem na fila para utilizar o microondas, ocorre um indeterminação de quem seria o próximo à usar o microondas, pois umas vez que Sheldon passar à frente de Howard e Howard à frente de Leonard e Leonard de Sheldon, pode-se observar um ciclo vicioso onde não se chega a conclusão de quem vai ser o próximo à usar o microondas. Sabendo disso, Raj a cada 5(cinco) segundos observa a fila para ver se está ocorrendo tal problema, caso o problema seja verdadeiro, Raj de forma imparcial escolhe uma pessoa da fila para usar o microondas encerrando assim o problema.

A função ”microondas” foi implementada como uma thread, onde vai consumir um personagem da fila a cada 1(um) segundo, e também existe uma função chamada ”add-personagem” que sua tarefa foi explicada no parágrafo anterior, assim como o problema que ela gera. Não levando em consideração Raj verificar se ocorreu um problema ou nao, acontece que ambas funções tem acesso a mesma região crítica (a fila de quem quer usar o microondas), e quando o função ”add-personagem” adiciona uma nova pessoa e causa o problema explicado anteriormente, por conta da utilização de semaforos do tipo mutex para exclusão mútua, faz com que a função ”micronondas” durma, sabendo que ”add-personagem” não irá acabar por causa do loop infinito, ela faz com que ”microondas” nunca tenha acesso a região crítica, esses casos em que uma funçao impede que outra nunca tenha acesso a região crítica é chamado de Starvation(Inanição).

IV. Conclusion

A aplicação práticas de ferramentas e conceitos aprendidos em sala como semáforos, threads, deadlock e starvation foram bem exercitadas nesse projeto. Utilizando uma exemplo abstrato, mais lúdico e que foge do contexto real no qual o cientísta da computação estudaria tal assunto, foi possível solidificar ainda mais o entendimento dos conceitos.

V. References

KRAMER, Jeff.”IS ABSTRACTION THE KEY TO COMPUTING?” COMMUNICATIONS OF THE ACM April 2007/Vol. 50, No. 4