Universidade de Brasília - Instituto de Ciências Exatas

Departamento de Ciência da Computação

Antônio Vinicius de Moura Rodrigues

Programação Concorrente:

Sincronização entre Processos

Brasília - 24 de março de 2022

2021.2

1. Introdução

O trabalho tem como objetivo apresentar o desenvolvimento de um sistema baseado na rotina de um hospital destinado a cuidar de pacientes com Coronavírus (COVID-19), generalizando o ambiente a apenas médicos, enfermeiras, pacientes e respiradores, que serão representados pela palavra “processo”, aplicando conceitos e métodos da programação concorrente para demonstrar um gerenciamento correto e efetivo no intuito de reduzir o nível de casos e mortes por essa doença.

2. Formalização do Problema Proposto

Avaliando a interação que há entre todos os processos, podemos avaliar uma grande interdependência, em que um processo precisa de trocar informações com outros para saber quando e como proceder.

Baseando se no problema proposto:

- **Hospital:** É onde está ambientado o algoritmo. Todos os processos estão entrelaçados a partir aqui;

- **Paciente:** Todos os processos funcionam a partir desse. O paciente decide se precisa ou não utilizar um respirador, ocupar um leito, consumir o remédio, e pedir alta ao final da internação;

- **Respirador:** É requisitada pelos pacientes, mas não por todos, alguns não necessitam utiliza-lo;

- **Enfermeiro:** Na falta de remédio, é encarregado pela reposição, e responsável por chamar o médico quando um paciente estiver pronto para receber a alta.

- **Médico:** Sua única função é assinar as altas dos pacientes que estão prontos para serem liberados.

3. Descrição do Algoritmo Desenvolvido

A troca de informações é um fator essencial para o bom funcionamento dos processos citados. O momento exato para receber uma mensagem a fim de saber quando prosseguir ou esperar são criados a partir do conceito de “semáforos”, onde um processo avalia se tem permissão para prosseguir ou não a partir de um ponto. Os semáforos criados neste algoritmo foram:

- **bed_count:** Contador de leitos, usado pelos pacientes para saber se pode ou não ocupar um leito;

- **lock_medicine:** Protege a variável “medicine” de ser acessada por mais de um processo simultaneamente, usada pelo enfermeiro quando está repondo os remédios e pelo paciente quando está usando;

- **lock_discharged:** Protege a variável “beds_used” de ser acessada por mais de um processo simultaneamente, usada pelo médico quando está liberando paciente e pelo paciente quando já pode ser liberado;

- **doctor_flag, nurse_flag, respirator_flag:** Acorda os processos ”Doctor”, “Nurse” e ”Respirator”, respectivamente, para realizarem o seu trabalho;

- **Free_respirador:** Libera o respirador que foi usado pelos pacientes quando já não precisam mais.

- **free_bed:** Sinaliza que um leito foi liberado, usado pelo paciente para avisar outros que já podem usar o local que ele liberou;

- **catch_respirator:** Sinaliza que o paciente está usando o respirador, é utilizado pelo respirador para avisar que está sendo usado pelo paciente e pelo paciente para esperar a liberação do respirador.

- **free_patient:** Sinaliza a alta do paciente, usada pelo médico para assinar a alta do paciente e pelo paciente para pedir alta.

- **free_medicine:** Sinaliza que a quantidade de remédios está sofrendo alteração, usado pela enfermeira durante a reposição e pelo paciente para esperar a reposição.

Também, foram utilizadas algumas variáveis:

- **beds_used:** Conta quantos pacientes pediram alta;

- **medicine_qty:** Quantidade de remédios disponível para consumo;

- **medicine:** Variável compartilhada entre enfermeira e pacientes para saber quando está faltando remédios;

- **necessary_medicine:** Quantidade de remédio que o paciente necessita;

- **need_bed:** Flag que sinaliza se há ou não leitos disponíveis.

- **need_respirator:** Variável que recebe um número, que é tratado a fim de saber se o paciente precisou ou não utilizar o respirador.

4. Conclusão

Conclui-se que é possível ambientar e simular um sistema utilizando os conceitos da programação concorrente, que utilizando a troca de informações conseguem chegar em um consenso para utilização adequada de uma memória compartilhada.

5. Referências

- Ben-Ari, Mordechai. “Principles of concurrent and distributed programming.” PHI Series in computer science (1990).
- Andrews, Gregory. “Concurrent programming - principles and practice.” (1991).
- Tanenbaum, Bos. “Sistemas operacionais modernos.” (2009)