Universidade de Brasília

Relatório Projeto Final
de
Programação Concorrente

Jean Rodrigues Magalhães
15-0079923

Introdução:

Na disciplina de programação concorrente, começamos a tratar e a desenvolver soluções para problemas concorrentes e que utilizam memória compartilhada. Tradando processos e utilizando Threads. Para tanto, conceitos como uso de Locks, semáforos e variáveis de condição foram apresentados. Cada aluno, no final do semestre, pôde escolher um tema e gerar uma problemática para que fosse solucionada utilizando os conceitos listados acima. O tema do meu projeto é “Emergência médica em tempos de COVID-19” e a problemática envolvida é no gerenciamento da chegada desses pacientes ao hospital, tratamento de superlotação e prioridade no atendimento de acordo com a urgência do seu caso.

Formalização do Problema Proposto:

Nesta emergência, dispomos de:

- Uma capacidade máxima de atendimento geral (pacientes que chegarem para serem atendidos e a emergência já tiver alcançado seu número máximo de pacientes, serão mandados para procurar atendimento em outro hospital).

- Um mecanismos de triagem para classificar a gravidade dos pacientes e direcionarmos os mesmos para o atendimento adequado (nessa triagem, temos apenas uma sala de atendimento e apenas um paciente por vez pode ser atendido, de acordo com as normas sanitárias).

- Sala de espera paras os consultórios médicos, na qual pacientes que chegam com maior gravidade não devem ficar aguardando na fila.

- 3 consultórios médicos para atendimento de paciente de pouca urgência e 1 consultório especial para atender paciente de muita urgência. Esporadicamente, alguns pacientes muito urgentes terão que utilizar consultórios normais, pois o especial está sendo utilizado por outro muito urgente (pacientes muito urgentes também terão prioridade no acesso aos consultórios não especiais).

- Tempos de consultas podem variar e consultórios podem atender mais pacientes que outros.

- Pacientes de muita Urgência são direcionados para intenação em UTI, após a sua consulta.

- Alguns pacientes pouco urgentes também precisaram de leitos normais de internação (seguindo dados aproximados de probabilidade de alguém precisar de internação, de acordo com o Ministério da Saúde).

- Se houver o esgotamento desses leitos, tanto de UTI quanto leitos normais, estes pacientes devem ser transferidos para outro hospital.

Descrição do Algoritmo Desenvolvido para Solução do Problema Proposto:

Para a solução, foram utilizados os seguintes raciocínios e técnicas:

- Na entrada do hospital, temos um semáforo para verificar a capacidade da emergência e se pacientes terão que ser dispensados.

```
sem_init(&sem_emergencia, 0, N_CAPACIDADE_EMERGENCIA);

if(sem_trywait(&sem_emergencia) != 0) {
  printf("Emergencia Cheia, paciente %d foi direcionado para outro hospital.\
n", id_paciente);
  pthread_exit(0);
 }
```

- Na triagem, também foi utilizado semáforo para garantir que apenas uma pessoa por vez fosse triada.

```
sem_init(&sem_sala_triagem, 0, 1);
```

- Para classificar os pacientes, foi utilizada uma função “geradora de status de gravidade”, para gerar Threads de pacientes muito urgentes ou pouco urgentes.

bool gera_gravidade_caso(bool gravidade);

- Pacientes têm seus resultados de triagem impressos na tela através da função:

void reposta_triagem(int id_paciente, bool gravidade);

- Eles são direcionados para a sala de espera dos consultórios e são chamados de um por um para o consultorio o próximo consultório disponivel, nessa parte tivemos que tratar a condições de corrida utilizando Locks nos consultorios:

pthread_mutex_t lock_consultorios = PTHREAD_MUTEX_INITIALIZER;

- Caso o paciente que acabou de sair da triagem seja muito urgente, ele não esperará na fila dos consultórios com os outros vários pacientes pouco urgentes.

```
if( gravidade == true){
    pthread_mutex_unlock(&lock_gravidade); //asseguramos que nenhum paciente grave concorra pela sala especial e entrem juntos no consultorio
                                           // se o consultorio especial ja estiver cheio, o novo paciente de muita urgencia vai ser o primeiro a ter acesso aos outros quando forem liberados

    consultorios_medicos(id_paciente, gravidade);// acesso prioritario aos consultorios pelos pacientes urgentes
 }
```

-Todos os pacientes recebem seus resultados das consultas. Para pacientes pouco urgentes, utilizamos mais uma vez de aproximações estatísticas para gerar uma quantidade de casos pouco urgentes que precisam vir a ser internados.

```
void resultado_consulta(int id_paciente){
  //de acordo com o Ministério da saúde em torno de 10% dos pacientes precisam ser hospitalizados
  int probabilidade_internacao = rand() % 100;

  // se estiver dentro desses 10% aleatorios, séra verificado a possibilidade de internação
  if (probabilidade_internacao >= 90) {
    printf("O paciente %d precisa ficar internado\n", id_paciente);
    verifica_leito( id_paciente, "normal"); //verifica se ainda há leitos neste hospital para pacientes com menos gravidade
  }else{
    //caso seja um paciente que possa continuar o tratamento em casa, ele é mandado pra casa e sua thread é encerrada
    printf("O paciente %d fará o tratamento em de recupeção da COVID-19 em casa\n", id_paciente);

  }
```

-Foram utilizados contadores (variáveis globais) para gerenciar o numero de leitos disponíveis nessa unidade e, assim, transferir os pacientes para outros Hospitais, caso o numero máximo de lotação de leitos fosse excedido. Esses tratamentos foram feitos com Locks.

-Todas as Threads são finalizadas após sabermos qual é o destino do tratamento daquele paciente ou para onde ele será alocado.

Conclusão

Podemos perceber que não precisamos procurar muito para encontrar um problema do nosso cotidiano que envolva situações que podem ser tratadas e resolvidas com os conceitos de Programação Concorrente. Nesta atividade, pude exemplificar e recriar de forma similar uma situação que vem atingindo nosso país e

o mundo todo. Mesmo com uma visão mais simplificada de um problema complexo como esse, podemos extrair informações importantes no comportamento de uma pandemia e como ela exige do seu sistema de saúde, no caso uma unidade de emergência com recursos físicos limitados para uma determinada demanda. Os resultados da simulação foram satisfatórios e, ao longo do seu desenvolvimento, podemos encontrar várias situações que se encaixariam naquele problema tanto para melhor representá-lo, quanto para propor soluções aos problemas que enfrentamos fora dos computadores.

Instruções para compilar e executar o trabalho:

cd “Diretório do arquivo”

gcc -pthread Simulation.c -o exec

./exec

Referências

• Aulas e slides do Professor Doutor da Universidade de Brasília Eduardo Adílio Pelinson Alchieri

• Ben-Ari, M., Principles of Concurrent and Distributed Programming, Prentice Hall, 2a ed., 2006.

• https://noticias.uol.com.br/saude/ultimas-noticias/redacao/2020/03/17/somente-1-em-cada-10-casos-do-novo-coronavirus-estao-hospitalizados.htm

• https://www.man7.org/linux/man-pages/man3/sem_init.3.html