LICENCIATURA EM ENGENHARIA INFORMÁTICA
UNIVERSIDADE DO MINHO

Sistemas Distribuídos

Exame[1]

13 de janeiro de 2025 Duração: 2h00m

Responda a cada grupo numa folha separada, entregando obrigatoriamente três folhas.

I

Responda diretamente a cada pergunta e com grafia bem legível. Omita preâmbulos e considerações genéricas sobre cada um dos assuntos.

1 Explique que tipo de situações justificam a utilização de uma variável de condição no controlo de programas concorrentes. Auxilie-se de um exemplo conciso em pseudo-código que suporte a sua resposta.

2 Considere a arquitetura genérica cliente/servidor e, no nó servidor, o atendimento orientado à conexão, com uma *thread* por conexão (*thread-per-connection*), e o atendimento orientado ao pedido, com uma *thread* dedicada a cada pedido (*thread-per-request*). Descreva sucintamente o que distingue os dois modelos e elabore sobre as vantagens de cada uma das abordagens.

3 Na implementação de um algoritmo distribuído para garantia de exclusão mútua no acesso a recursos partilhados, argumente sobre as vantagens e desvantagens da utilização de relógios de tempo real (*hardware clocks*) ou lógicos, como o proposto por Leslie Lamport.

4 Qual o problema endereçado pelo protocolo de *two-phase-commit*? Elabore sobre as limitações do algoritmo, se as houver, no caso de inexistência de falhas dos nós participantes

II

Considere um sistema de gestão de acessos a galerias de um museu. O museu é composto por N galerias, com identificadores de 1 a N. Cada galeria tem uma capacidade máxima de C pessoas. Uma pessoa que pretenda visitar uma ou mais galerias adquire primeiro um bilhete. Na compra do bilhete, indica ao sistema o número de galerias que quer visitar. Quando uma pessoa tenta entrar numa galeria, poderá ter que ficar à espera até que saia alguém, caso o limite C já tenha sido atingido. O sistema deverá assegurar que uma pessoa não consegue entrar numa galeria com um bilhete inválido (por exemplo, se o cliente adquirir um bilhete que permite visitar três galerias, a tentativa de uma quarta entrada com esse bilhete deverá ser negada). O cliente poderá ainda consultar o número de pessoas à espera de entrar em cada galeria a qualquer momento.

Apresente uma classe Java (para ser usada no servidor), usando primitivas baseadas em monitores, que implementa a interface abaixo, tendo em conta que os seus métodos serão invocados num ambiente *multi-threaded*.

```
interface MuseumManager {
    String buyTicket(int uses);
    int enterGallery(int galleryId, String ticketId) throws InterruptedException;
    void exitGallery(int galleryId, String ticketId);
    Map<Integer, Integer> peopleWaitingPerGallery();
}
```

Funcionalidade básica (até 80%): O método `buyTicket` é usado para comprar um bilhete, onde `uses` é o número de galerias que a pessoa quer visitar, retornando um identificador único para o bilhete. O método `enterGallery` é usado para entrar numa galeria, recebendo os identificadores da galeria e do bilhete. Caso o bilhete seja válido, deve esperar até que haja espaço na galeria, retornando depois o número da galeria; caso o bilhete seja inválido, deve retornar 0. O método `exitGallery` serve para indicar a saída da pessoa de uma galeria, recebendo os identificadores da galeria e do bilhete. O método `peopleWaitingPerGallery` retorna o número de pessoas à espera de entrar, para cada galeria. Procure maximizar o paralelismo permitido e minimizar o número de *threads* que são acordadas.

Funcionalidade avançada (para 100%): Admita que o método `enterGallery` pode receber 0 no parâmetro `galleryId`. Nesse caso, deve escolher uma galeria que não tenha sido visitada, usando esse bilhete, com espaço livre. Se todas estiverem cheias, deve bloquear até uma delas ficar com espaço livre.

III

Considere o sistema descrito acima, ao qual clientes se ligam por TCP. Implemente só o programa servidor usando *threads*, *sockets* TCP, e a interface apresentada na pergunta anterior. Descreva o protocolo usado, que deve ser o mais simples possível, por exemplo, baseado em linhas de texto.

[1] Cotação — 10+7+3