Nome: _____ **Número:** _____

LICENCIATURA EM ENGENHARIA INFORMÁTICA

Sistemas Distribuídos

Exame de Época Especial

16 de Setembro de 2011 Duração: 1h30m

I

1 O monitor é uma primitiva estruturada de controlo de concorrência que consideramos de "alto nível". Comente esta afirmação e indique uma vantagem e uma desvantagem deste elevado nível de abstracção.

Um monitor é uma primitiva estruturada do controlo de concorrência. Oferece um tipo de dados com controlo de concorrência implícito em todas as operações de exclusão mútua. Com um modelo de concorrência baseado em monitores, o compilador pode inserir mecanismos de exclusão mútua transparente, isto é, sem o programador ter que aceder às primitivas de controlo e realizar o bloqueio e libertação de recursos manuais.
Em java a utilização de monitores, é realizada através de métodos e blocos synchronyzed que fazem uso de ReentrantLocks.

Vantagens: Controlo de concorrência implícita (transparente);

Desvantagens: Os locks são reentrantes e podem levar à starvation.

2 No paradigma cliente/servidor caracterize, de forma genérica, as funções atribuídas aos clientes e aos servidores.

Cliente: - Interface com o utilizador;
- Caching;

Servidor: - grande capacidade de armazenamento e/ou processamento;
- perifericos especiais : ex: impressão, salvaguarda;
- perifericos particulares : ex: pesquisa, autenticação;
- exemplos de servidores vulgares: ficheiros, base de dados, informação de redes, nomes, email, autenticação.

Há dois tipos de servidores, o statefull e o stateless.
O servidor stateless diz-se sem estado se de uma invocação para a seguinte este não mantém qualquer informção acerca o estado do cliente, ou seja, não sabe se o cliente guarda algo em cache. Neste caso o cliente é o unico responsavel pelas suas acções. Um servidor stateless executa uma operação e esquece o cliente: Recupera bem após um crash, tipicamente precisa de mais argumentos para cada invocação, o estado esta no cliente e é passado ao servidor. ex: web proxies, web servers, NFS.
O sevidor statefull tem todas as informações sobre os clientes: pode ser mais eficiente, pois tem mais informaçao, é complicado recuperar apõs um crash, estado em memoria persistente demora a atualizar e a recuperar.

II

Considere um servic¸ o distribú ıdo de presenc¸ as em que os seus utilizadores (identificados por nome) registam sempre a sua entrada num determinado espaço (também identificado por nome). Cada espaço é controlado por um servidor local específico. Opcionalmente, os utilizadores podem registar a saída desse espaço no referido servidor. Os utilizadores podem, portanto, entrar, migrar ou abandonar espaços controlados pelo sistema de registo de presença. Além das operações de entrada e saída de utilizadores, os servidores devem ainda suportar a operação de localização de um utilizador informando qual o id do espaço (e respectiva data e hora de entrada) onde poderá ser encontrado.

Implemente o servidor de presença descrito assumindo que recebe como argumentos da sua linha de comando, o id do espaço que deverá controlar, o porto de escuta e os endereços de todos os servidores do sistema. Assuma que todos os servidores são reciprocamente acessíveis, e a data e hora local estão suficientemente sincronizadas. Recorra a `System.currentTimeMillis()` para obter a data e hora expressa em milisegundos. Procure maximizar a concorrência potencial da sua solução.

```
$ servidor <espaco-id> <porto-ip> <endereco-ip-serv-1> ... <endereco-ip-serv-n>
```

Algumas primitivas relevantes

- ```
  Object o;
  synchronized(o) { ...  };
  o.wait();
  o.notify();
  o.notifyAll();
  ```

- ```
  Lock l = new ReentrantLock();
  Condition c = l.newCondition();
  l.lock();
  l.unlock();
  c.await();
  c.signal();
  c.signalAll();
  ```