Nome: Tânia Rocha MIEI **Número:**

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
UNIVERSIDADE DO MINHO

Sistemas Distribuídos

Teste[1]

9 de Janeiro de 2017 Duração: 2h00m

I

1 Distinga, em termos de objetivo e forma de utilização, as primitivas de *lock/unlock* e *wait/notify*.

O objetivo das primitivas lock/unlock é garantir que apenas um processo se encontra na zona envolvida por estas primitivas. O objetivo das primitivas wait/notify é que os processos voluntariamente adormeçam enquanto o predicado de uma variável de condição não é verificada, sendo acordados por outros processos quando esses alteram algo e existe a possibilidade de o predicado ser válido.
A forma de utilização de lock e unlock é basicamente adquirir o lock(lock), fazer o que é necessário e depois libertar(unlock).
A forma de utilização de wait/notify está normalmente associada a ter um lock para verificar um predicado e caso se verifique, liberta-se o lock, adormecendo(wait) sendo que quando for acordado terá de adquirir o lock. Quando "passa"o predicado liberta o lock. Depois de realizar o que tem de realizar o que tem de realizar adquire novamente o lock, pois vai mexer em algo partilhado e consoante aquilo que faz pode notificar quem está à espera naquela condição libertando o lock de seguida.

2 Das arquitecturas de sistemas distribuídos que estudou qual ou quais se adequariam melhor a um sistema de suporte a redes sociais?

As redes sociais são baseadas em eventos/publicações as quais podem gerar reacções em determinados componentes. É exatamente neste contexto que foram criadas as arquiteturas baseadas em eventos que a meu ver são as que melhor se adequam a um sistema de suporte deste tipo de ambiente.
Este tipo de arquitetura é constituído por um grande repositório de eventos(dados), dados estes que são homogéneos, orientados a uma aplicação em tempo real. Trata-se de um barramento de eventos e vários componentes que de forma assíncrona(não esperando por um determinado evento) agem perante estes. Tal comportamento é identificável em qualquer rede social atual.
Neste tipo de arquitetura, os processos publicam eventos e o middleware assegura que apenas os processos que se subscreveram (“se inscreveram”) para esses eventos os receberão;

3 Qual a relevância do sistema operativo na resolução do problema de exclusão mútua no modelo de memória partilhada e no modelo de passagem de mensagens?

A função do sistema operativo na resolução do problema de exclusão mútua tem por base uma eficiente gestão dos recursos. Este é responsável por bloquear processos, prevenindo-os de consumir tempo de CPU, enquanto não tiverem permissão para avançar para a região crítica.
A ação do sistema operativo neste tipo de problemas no modelo de memória partilhada e no de passagem de mensagens apenas difere no momento em que os processos são bloqueados e libertados.
Em memória partilhada tal acontece quando se tenta obter o lock, no caso da passagem de mensagens, os processos são bloqueados desde que enviam os pedido até à receção da resposta.
Em ambos os casos evitam-se as esperas ativas

[1] Cotação — 10+10

II

Considere um serviço simplificado de controlo de aquecimento numa casa (exemplo de *internet of things*) onde os equipamentos se ligam por TCP/IP a um servidor. Existem os seguintes dispositivos: Um termómetro que se liga para indicar a **temperatura** atual da casa; Um controlo de temperatura alvo que se liga para indica o **limiar** de activação da caldeira; Um relé que controla a caldeira e que indica o seu estado actual (off = false ou on = true) e aguarda indicação de mudança de estado via o método **on_off**. Os métodos **temperatura** e **limiar** devem retornar imediatamente, mas o método **on_off** deve bloquear caso o estado fornecido seja idêntico ao estado actual do sistema. Por exemplo, se a temperatura actual for de 16º e o limiar actual for de 19º então o sistema sabe que a caldeira deve estar ligada para aquecer a casa; se o relé da caldeira invocar `on_off(true)`, indicando já estar ligado, então o método só deve retornar (indicando **false**) quando a temperatura ultrapassar os 19º.

```
interface Controlador {
  temperatura(int centigrados);        // medidor indica temperatura actual
  limiar(int centigrados);             // utilizador indica limiar de activação
  boolean on_off(boolean estadoatual); // caldeira pergunta se estado mudou
}
```

1 Apresente uma classe que implemente o interface Controlador tendo em conta que os seus métodos poderão ser invocados num ambiente multi-threaded.

2 Implemente um servidor em rede usando sockets TCP que disponibilize os métodos da classe desenvolvida na pergunta anterior. O servidor deverá suportar a conexão de múltiplos clientes em simultâneo.