Sistemas Distribuídos
Teste 1

11 de Junho de 2007

Parte A (12 valores)

1. Explique o que entende por *sistema distribuído*. Dê dois exemplos da sua aplicação sobre plataformas hardware com características distintas. Qual é a motivação da sua construção em cada um dos casos?
2. Descreva esquematicamente, introduzindo os comentários apropriados sobre a sua operacionalidade, vantagens e inconvenientes, as três variantes do modelo cliente-servidor que foram estudadas.
3. No contexto do RMI em Java, explique o que entende por objectos [remotos] estáticos e dinâmicos. Descreva funcionalmente como é que eles podem ser implementados e dê um exemplo de cada tipo da sua aplicação.
4. No âmbito da comunicação entre pares, descreva detalhadamente um algoritmo que permite de uma forma distribuída e dinâmica a um dado processo assumir-se como líder. Indique claramente quais são os pressupostos que estão subentendidos.

Parte B (8 valores)

```
public class Semaphore
{
   private int val = 0,
   numbBlockThreads = 0;
   public synchronized void down ()
   {
      if (val == 0)
         { numbBlockThreads += 1;
           try
           { wait ();
           }
           catch (InterruptedException e) {}
         }
         else val -= 1;
   }
   public synchronized void up ()
   {
      if (numbBlockThreads != 0)
         { numbBlockThreads -= 1;
           notify ();
         }
         else val += 1;
   }
}

public class StoreRegion1
{
   private int mem = 0;
   private Semaphore [] stat = new Semaphore[4];
   private StoreRegion2 store = null;
   public StoreRegion1 (StoreRegion2 store)
   {
      for (int i = 0; i < 4; i++)
        stat[i] = new Semaphore ();
      this.store = store;
   }
   public int getInfoP ()
   {
      stat[2].up ();
      stat[0].down ();
      return mem;
   }
   public int getInfoC ()
   {
      stat[3].up ();
      stat[1].down ();
      return mem;
   }
   public void processInfo ()
   {
      stat[2].down ();
      stat[3].down ();
      int nBuff = store.getBuff ();
      mem = nBuff;
      stat[0].up ();
      stat[1].up ();
   }
}
```

```
public class StoreRegion2
{
   private int size = 0;
   private boolean [] pSit = null;
   private int [] mem = null;
   private Semaphore [] stat;
   public StoreRegion2 (int size)
   {
      this.size = size;
      pSit = new boolean[size];
      for (int i = 0; i < size; i++)
        pSit[i] = false;
      mem = new int[size];
      stat = new Semaphore[2*size+1];
      for (int i = 0; i < 2*size+1; i++)
        stat[i] = new Semaphore ();
      for (int i = 0; i < size; i++)
        stat[0].up ();
      for (int i = 1; i < size+1; i++)
        stat[i].up ();
   }
   public void putVal (int nBuff, int val)
   {
      stat[nBuff+1].down ();
      mem[nBuff] = val;
      stat[nBuff+size+1].up ();
   }
   public int getVal (int nBuff)
   {
      stat[nBuff+size+1].down ();
      int val = mem[nBuff];
      stat[nBuff+1].up ();
      return val;
   }
   public int getBuff ()
   {
      int nBuff = -1;
      stat[0].down ();
      synchronized (this)
      { for (int i = 0; i < size; i++)
          if (!pSit[i])
             { nBuff = i;
               pSit[i] = true;
               break;
             }
      }
      return nBuff;
   }
   public void relBuff (int nBuff)
   {
      synchronized (this)
      { pSit[nBuff] = false;
      }
      stat[0].up ();
   }
}

public class ThreadType1 extends Thread
{
   private int id;
   private StoreRegion1 store1 = null;
   private StoreRegion2 store2 = null;
   public ThreadType1 (int id, StoreRegion1 store1, StoreRegion2 store2)
   {
      this.id = id;
      this.store1 = store1;
      this.store2 = store2;
   }
   public void run ()
   {
      int [] val = {3, 2, 1, 4};
      int nBuff;
```

```
      for (int i = 0; i < val.length; i++)
      { try
        { sleep ((int) (1 + 100*Math.random ()));
        }
        catch (InterruptedException e) {};
        nBuff = store1.getInfoP ();
        store2.putVal (nBuff, 100*id+val[i]);
      }
   }
}

public class ThreadType2 extends Thread
{
   private int id,
               niter;
   private StoreRegion1 store1 = null;
   private StoreRegion2 store2 = null;
   public ThreadType2 (int id, int niter, StoreRegion1 store1, StoreRegion2 store2)
   {
      this.id = id;
      this.niter = niter;
      this.store1 = store1;
      this.store2 = store2;
   }
   public void run ()
   {
      int val;
      int nBuff;
      for (int i = 0; i < niter; i++)
      { try
        { sleep ((int) (1 + 100*Math.random ()));
        }
        catch (InterruptedException e) {};
        nBuff = store1.getInfoC ();
        val = store2.getVal (nBuff);
        System.out.println ("O valor produzido por P" + (val/100) + " e por C" +
                            id + " no buffer " + nBuff + " foi " + (val%100) + ".");
        store2.relBuff (nBuff);
      }
   }
}

public class ThreadType3 extends Thread
{
   private int niter;
   private StoreRegion1 store = null;
   public ThreadType3 (int niter, StoreRegion1 store)
   {
      this.niter = niter;
      this.store = store;
   }
   public void run ()
   {
      for (int i = 0; i < niter; i++)
        store.processInfo ();
   }
}

public class SimulSituation
{
   public static void main (String [] args)
   {
      StoreRegion2 store2 = new StoreRegion2 (2);
      StoreRegion1 store1 = new StoreRegion1 (store2);
      ThreadType1 [] thr1 = new ThreadType1[4];
      for (int i = 0; i < 4; i++)
        thr1[i] = new ThreadType1 (i+1, store1, store2);
      ThreadType2 [] thr2 = new ThreadType2[2];
      for (int i = 0; i < 2; i++)
        thr2[i] = new ThreadType2 (i+1, 8, store1, store2);
      ThreadType3 thr3 = new ThreadType3 (16, store1);
      thr3.start ();
```

```
for (int i = 0; i < 2; i++)
        thr2[i].start ();
      for (int i = 0; i < 4; i++)
        thr1[i].start ();
    }
}
```

1. Representando as entidades activas por círculos e as entidades passivas por rectângulos, faça um diagrama ilustrativo da interacção em presença e indique por palavras simples qual é o papel desempenhado pelas *threads* de cada tipo (não mais do que uma frase).
2. Explique qual é o significado de cada um dos elementos do *array* `Semaphore` dos tipos de dados `StoreRegion1` e `StoreRegion2`.
3. Altere o programa de modo a ser possível comunicar pares de valores entre os *threads* de tipo 1 e tipo 2.