29/01/2004

Ejercicio 1.- Analice el siguiente código y responda a las siguientes preguntas, justificando las respuestas.

a) ¿Qué misión tienen, el, o los, sockets que aparecen? ¿Con qué constructor específico se construyen?

b) La clase chargenURLConnection ¿De qué tipo es? ¿Qué características específicas tiene?

```
import java.net.*;
import java.io.*;
public class chargenURLConnection extends URLConnection  {
        Socket theConnection = null;
        public final static int defaultPort = 19
        public chargenURLConnection (URL u) {
                super (u) ;
```

```
        }
public synchronized InputStream getInputStream() throws IOException {
        if (!connected) {
        connect( ) ;
        }
        return theConnetion.getInputStream();
        }
        public String getContentType() {
                return text/plain;
        }
        public synchronized void connect() throws IOException {
                int port;
                if (!connected) {
                        port = url.getPort();
                        if (port < 0) {
                                port = defaultPort;
                                }
                                theConnection = new Socket(url.getHost(), port);
                                connected = true;
                                }

}
```

Solución.- La clase chargenURLConnection es una subclase de la clase URLConnection. Esta clase hereda los métodos getContentType() y getInputStream() de URLConnection e implementa connect(). Tiene también un constructor que construye un nuevo URLConnection desde un URL.

La clase tiene dos campos, el socket theConnection entre el cliente y el servidor. Los métodos getInputStream( ) y connect( ) necesitan acceder a este campo, por lo que no puede ser una variable local. El segundo campo es defaultPort, un entero estático final, que contiene el puerto por defecto del protocolo chargen; este puerto se utiliza si no se cita de forma explicita.

El constructor de la clase no presenta sorpresas. Llama al constructor de la superclase con el mismo argumento, el URL u. El método connect( ) abre una conexión al servidor especificado sobre el puerto especificado (o si no se ha especificado entonces por el puerto por defecto de chargen, 19); y asumiendo que la conexión se ha efectuado, define la variable booleana connected a true. Recordamos que connected es un campo protegido en java.net.URLConnection que se hereda por la subclase. El Socket que abre connect( ) está almacenado en el campo theConnection para poder usarlo más tarde mediante getInputStream. El método connect está synchronized para evitar una posible falta de condición de exclusión mutua sobre la variable connected.

El método getContentType() devuelve un String conteniendo un tipo MIME para el dato. Este es utilizado por el método getContent() de java.net.URLConnection para seleccionar el manejador de contenido apropiado. El dato devuelto por un servidor de chargen es siempre texto ASCII, puesto que el método getContentType() devuelve text/plain. El método getInputStream devuelve un InputStream desde el Socket que crea connect. Si no se establece la conexión cuando se llama a getInput Stream( ), el método llama por si mismo a connect.

Ejercicio 2.- Analice las siguientes afirmaciones y justifique si son o no ciertas.

a) Desde un punto de vista lógico un S.D. es un conjunto de procesos que ejecutan en una o más computadoras y que se comunican y sincronizan mediante paso de mensaje.
b) Un hilo tiene un mejor rendimiento en cuanto a eficiencia que un proceso.
c) La interfaz UICI implementa una comunicación cliente-servidor orientada a conexiones.
d) El problema de la Sección Crítica en un sistema distribuido se resuelve mediante semáforos o monitores.

Solución

a) Es cierta, la comunicación es por lo general por paso de mensajes.
b) Depende, dado que al compartir el mismo espacio de direcciones, las variables globales y el mismo conjunto de archivos abiertos su tiempo de procesamiento es menor, pero siempre que utilicemos una misma máquina, en otro caso es mejor utilizar procesos.

c) Es cierta, UICI es la Universal Internet Communication Interface, la comunicación esta orientada a conexiones mediante sockets, TLI o STREAMS.
d) No es cierta, el problema de SC se resuelve mediante algoritmos específicos como el del anillo, de Ricart y Agrawala o el centralizado.

Ejercicio 3.- a) Algoritmo del anillo de recuperación de fallo de coordinador. Ventajas e inconvenientes sobre otros algoritmos que resuelvan el mismo problema.
b) Implante utilizando primitivas de pvm, el algoritmo del anillo para que una serie de procesos se recuperen del fallo del coordinador, a partir del siguiente código. (1) Explique el sentido y la funcionalidad de las primitivas utilizadas.

Solución

Suponemos que los procesos tienen un orden total, físico o lógico. Cuando algún proceso observa que el coordinador no funciona, construye un mensaje ELECCIÓN con su propio número de proceso y envía el mensaje a su sucesor. Si este está inactivo, pasa el mensaje al siguiente elemento del anillo, hasta que localiza un proceso en ejecución. En cada paso, el proceso emisor añade su propio número de proceso a la lista en el mensaje. Cuando el mensaje es recibido por un proceso que detecta que en el mensaje está su número de proceso, pasa un nuevo mensaje COORDINADOR y lo hace circula de nuevo pero ahora pasando en este mensaje como nuevo coordinador aquel proceso con el número mayor de la lista recibida y quienes son los miembros del nuevo anillo. Una vez que el mensaje ha pasado a todos los procesos del anillo termina el algoritmo de elección.

La desventaja mayor está, en que es posible, que uno o varios procesos detecten a la vez el fallo del coordinador y construyan un mensaje de ELECCIÓN y lo hagan circular, cuando los mensajes den una vuelta completa inician los mensajes COORDINADOR y cuando den una vuelta completa se eliminan. Si es un solo proceso el que detecta el fallo del coordinador y hay n procesos en el anillo el número de mensajes es de 2n, n de ELECCIÓN y n de COORDINADOR, presenta en general menos mensaje que el algoritmo del "más grande" de García Molina

Código de construcción de un anillo

```
#define NPROC 10
#include <stdio.h>
#include <sys/types.h>
#include "pvm3.h"

main()
{
   int mytid;
   int tids[NPROC];
   int me;
   int i;
   mytid = pvm_mytid();
   me = pvm_joingroup("nombre");
   printf("me = %d mytid = %d\n",me,mytid);
   if( me == 0 )
   pvm_spawn("eleccion", (char**)0, 0, "", NPROC-1, &tids[1]);
   pvm_freezegroup("nombre", NPROC);
   pvm_barrier("nombre", NPROC);
   generador(me, NPROC);
   pvm_lvgroup("nombre");
   pvm_exit();
   exit(1);
}
```

```
generador(me, NPROC)
    int me;
    int NPROC;
{
    int eleccion;
    int src, dest;
    int count  = 1;
    int stride = 1;
    int msgtag = 4;
    src = pvm_gettid("nombre", me-1);
    dest= pvm_gettid("nombre", me+1);
    if(me == 0)       src = pvm_gettid("nombre",
NPROC-1);
    if(me == NPROC-1) dest = pvm_gettid("nombre
0);
    if(me == 0)     {
      eleccion = dest;
      pvm_initsend(PvmDataDefault);
      pvm_pkint(&eleccion, count, stride);
      pvm_send(dest, msgtag);
      pvm_recv(src, msgtag);
      printf("fin \n");
    }
    else
    {
      pvm_recv(src, msgtag);
      pvm_upkint(&token, count, stride);
      pvm_initsend(PvmDataDefault);
      pvm_pkint(&eleccion, count, stride);
      pvm_send(dest, msgtag);
    }
}
```