Mensajes numéricos UDP entre C++ y JAVA

Elaborado por: Ukranio Coronilla

Al código servidor del capítulo 9 del manual "***Programación de Sistemas Linux***" hágale modificaciones para que imprima los dos enteros que recibe y coméntele la función `sendto()`. Ejecútelo en una terminal.

En otra terminal compile y ejecute el siguiente cliente en Java con nombre `udp_cliente.java`

```
import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;  //Para manejo de ByteBuffer

public class udp_cliente
{
    public static void main(String args[])
    {
        DatagramSocket sock = null;
        int port = 7200;

        try
        {
            sock = new DatagramSocket();
            InetAddress host = InetAddress.getByName("localhost");

            final ByteBuffer buf = ByteBuffer.allocate(8); // Reserva 8 bytes para 2 enteros
            buf.putInt(-2147483648);    //Mínimo valor entero
            buf.putInt(2147483647);     //Máximo valor entero
            DatagramPacket  dp = new DatagramPacket(buf.array() , buf.limit(), host , port);
            sock.send(dp);
        }
        catch(IOException e)
        {
            System.err.println("IOException " + e);
        }
    }
}
```

Como podrá observar en el código de Java se utiliza la clase **ByteBuffer** con objeto de acceder a los bytes con ayuda de los métodos descritos en:

http://docs.oracle.com/javase/7/docs/api/java/nio/ByteBuffer.html

También se puede dar cuenta que aunque se envían los enteros -2147483648, y 2147483647, en el servidor se imprimen otros enteros. Esto se debe a que **ByteBuffer** almacena los bytes en el buffer en formato big endian por default.

Ejercicio 1 : Corrija en el cliente para que los números se impriman correctamente en el servidor. Sugerencia: Hacer los cambios en el cliente para enviar datos en formato little endian con ayuda del método `order` incluido en `java.nio.ByteOrder`

Ejercicio 2: De la misma forma que los métodos `putInt` en el código del cliente, utilice los métodos `putDouble`, `putFloat` y por último `putLong` para enviar los valores Máximos y mínimos permitidos en cada uno de los tipos de datos correspondientes en un solo programa. Se recomienda verificar siempre el tamaño de cada tipo de dato en ambos lenguajes (en Java no existe la función `sizeof(int)`, pero se puede imprimir `Integer.SIZE`). Sugerencia: Para ver los valores máximos y mínimos en Java puede ejecutar el siguiente código en Java:

```
System.out.println("Tipo\tMínimo\tMáximo");
System.out.println("byte\t" + Byte.MIN_VALUE + "\t" + Byte.MAX_VALUE);
System.out.println("short\t" + Short.MIN_VALUE + "\t" + Short.MAX_VALUE);
System.out.println("int\t" + Integer.MIN_VALUE + "\t" + Integer.MAX_VALUE);
System.out.println("long\t" + Long.MIN_VALUE + "\t" + Long.MAX_VALUE);
System.out.println("float\t" + Float.MIN_VALUE + "\t" + Float.MAX_VALUE);
System.out.println("double\t" + Double.MIN_VALUE + "\t" + Double.MAX_VALUE);
```

Pregunta 1: ¿Puesto que puedo almacenar números más grandes en un float de 4 bytes que en un long de 8 bytes, entonces me conviene siempre usar float? ¿En qué situaciones es más conveniente uno, y en cuales otro?

Ejercicio 3: Repita el ejercicio 2, pero ahora con el cliente en C++ y el servidor en Java. Para ello puede ocupar el siguiente código Java para el servidor:

```
byte[] buffer = new byte[65536];
DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
sock.receive(reply);

byte[] data = reply.getData();
ByteBuffer buf = ByteBuffer.wrap(data);
buf.order(ByteOrder.LITTLE_ENDIAN);

int entero1 = buf.getInt();
int entero2 = buf.getInt();
System.out.println("int MIN = " + entero1 + " int MAX = " + entero2);
```

Ejercicio 4: Ahora intente enviar la siguiente estructura (inicializada con los valores más grandes que admita cada variable) de un código C++ hacia Java mediante el socket UDP. Sugerencia: Saque el tamaño de la estructura para determinar las posibles fallas en el envío.

```
struct mensaje{
   int entero;
   double doble;
   float flotante;
   long largo;
}
```

Ejercicio 5: Envíe ahora la misma información del ejercicio anterior pero desde Java y recíbalo en la estructura `struct mensaje`. Recuerde que Java no admite estructuras.