Segundo Certamen: Tiempo: 11:45 hrs - 13:30 hrs.
Todas las preguntas tienen igual puntaje.

Prepare un archivo llamado nombre.apellido.doc (o .odt) en él ponga cada una de sus respuestas.
P1: Respuesta
P2: Respuesta
P3: Respuesta
Además puede adjuntar un archivo nombre.apellido.tar con los programas de sus respuestas.

Considere un mini servidor multicast TCP de capa aplicación como el mostrado en la figura. Este servidor se ejecuta con el puerto de escucha como su único parámetro. Una vez corriendo, él espera y acepta tres conexiones. Conexiones posteriores son ~~son~~ rechazadas. Este servidor replica todo lo que llega por cualquiera de sus conexiones en las otras conexiones (una o dos según las haya). Por pantalla el servidor muestra la IP y puerto de origen del mensaje además del mensaje enviado; como en:
200.1.17.17:34567 Hola a todos.

Nota: Para probarlo se puede usar telnet <máquina_servidora> <puerto>

1.- Usando "select" programe un mini servidor multicast TCP. El servidor se ejecuta:
$ miniServidor <puerto>
Su código debe excluir sentencias y variables no relacionadas con lo pedido.
En otras palabra: si usa códigos vistos en clases, debe limpiarlos.

```
#include <stdio.h>  /* printf */
#include <stdlib.h>  /*exit*/
#include <string.h> /* memcpy*/
#include <netdb.h>

void reusePort(int sock);

int main( int argc, char *argv[] ) {
  int s;
  struct sockaddr_in server;
  struct sockaddr_in from[3];
  int fromlen;
  int length;
  fd_set readfds, readfdsCopy;
  int n,i;
  int sockList[3], lastSock=0;

  server.sin_family = AF_INET;
  server.sin_addr.s_addr = htonl(INADDR_ANY);
  server.sin_port = htons(atoi(argv[1]));
  s = socket (AF_INET,SOCK_STREAM,0);
  reusePort(s);
  if ( bind( s, (struct sockaddr *)&server, sizeof(server) ) ) {
    close(s);
    perror("binding name to stream socket");
    exit(-1);
  }
  listen(s,4);
  fromlen = sizeof(from);
  FD_ZERO(&readfdsCopy);
```

```
  FD_SET(s,&readfdsCopy);
  for(;;){
    memcpy(&readfds, &readfdsCopy, sizeof(fd_set));
    n = select(FD_SETSIZE, &readfds, (fd_set *) 0, (fd_set *) 0, NULL);
    if (n > 0) {
      if (FD_ISSET(s, &readfds)) {
        printf("Accepting a new connection...\n");
        sockList[lastSock] = accept(s, (struct sockaddr *)&(from[lastSock]),
                                    &fromlen);
        FD_SET(sockList[lastSock], &readfdsCopy);
        lastSock++;
      }
      for (i=0; i < lastSock; i++) {
        if (FD_ISSET(sockList[i], &readfds))
          if (distribuyaMensaje(sockList, i, from, lastSock) < 0) {
            FD_CLR(sockList[i], &readfdsCopy);
            close (sockList[i]);
            sockList[i] = sockList[--lastSock];
            i--;
            printf("Number of current clients: %d\n", lastSock);
          }
      }
    }
  }
}

int distribuyaMensaje(int *sockList, int i, struct sockaddr_in *from, int lastSock){
  char buf[512];
  int rc,j;

  if( (rc=read(sockList[i], buf, sizeof(buf))) < 0)
    perror("receiving stream  message");
  if (rc > 0){
    buf[rc]='\0';
    printf("%s:%d  %s\n", inet_ntoa(from[i].sin_addr),
          ntohs(from[i].sin_port), buf);
    for (j=0; j<lastSock; j++)
      if (j!=i)
        if (send(sockList[j], buf, rc, 0) <0 )
          perror("sending stream message");
    return(1);
  }
  else {
    printf("Disconnected..\n");
    return(-1);
  }
}
void reusePort(int s){
  int one=1;

  if ( setsockopt(s,SOL_SOCKET,SO_REUSEADDR,(char *) &one,sizeof(one)) == -1 ){
    printf("error in setsockopt,SO_REUSEPORT \n");
    exit(-1);
  }
}
```

Es importante el buen uso de select y mecanismo para distribuir el mensaje. No hay problema de sincronismo pues se trata de sólo un proceso e hilo.

2.- Usando hilos programe el mini servidor multicast TCP en Java. El servidor se ejecuta:
$ java miniServidor <puerto>
Su código debe excluir sentencias y variables no relacionadas con lo pedido.

```
import java.io.*;
import java.net.*;

public class MiniServidor {

  public static void main (String args[]) {
    try{
      ServerSocket s = new ServerSocket(Integer.parseInt(args[0]));
      ArraySocket as= new ArraySocket();
      for (int i=0; i<3; i++) {
        Socket cs= s.accept();
        as.add(cs);
        Client client = new Client(cs, as);
        client.start();
      }
    }  catch (Exception e) {
      e.printStackTrace();
    }
  }
}

class ArraySocket {
  Socket sockets[];
  int lastSocket;

  public ArraySocket (){
    sockets= new Socket[3];
    lastSocket=0;
  }
  public synchronized void add (Socket s){
    sockets[lastSocket++]=s;
  }
  public synchronized void send (String str, Socket excluir){
    System.out.println(excluir.getInetAddress().getHostAddress()+":"+
                       excluir.getPort()+"  "+str);
    try {
      for (int i=0; i< lastSocket; i++)
        if (sockets[i]!=excluir)
          new PrintWriter(sockets[i].getOutputStream(), true).println(str);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public synchronized void remove(Socket s){  // esta implementación no se requería
    int i;
    for (i=0; sockets[i]!=s && i<lastSocket; i++);
    if (i<lastSocket) sockets[i]=sockets[--lastSocket];
  }
}

class Client extends Thread
{
  ArraySocket as;
  Socket cs;
```

```
  public Client (Socket s, ArraySocket a){
    cs = s;
    as = a;
  }

  public void run () {
    boolean done=false;
    try{
      BufferedReader in = new BufferedReader
        (new InputStreamReader(cs.getInputStream()));
      while (!done) {
        String line = in.readLine();
        if (line == null) done = true;
        else
          as.send(line, cs);
      }
      as.remove(cs);
      cs.close();
    }  catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

3.- En este problema usted debe mostrar cómo acceder desde Java a códigos pre-existentes escritos en lenguaje C.
Se cuenta con el archivo suma.o, el cual es la versión compilada de la implementación de la función cuyo prototipo es:
int suma(int a, int b); /* retorna la suma a+b */
Desarrolle un programa en Java que retorna la suma de dos enteros pasados como argumentos de la ejecución. Usted debe usar JNI (Java Native Interface) para obtener la suma de los valores usando implementación en suma.o.

Si la ejecución es:
$java SumaJava 20 34
El valor mostrado por Java en consola será 54.

Muestre todos los comandos a ejecutar hasta crear su biblioteca de ligado dinámico.
Nota: El código nativo a programar por usted debe invocar a suma() para realizar lo pedido.

Programa Java:

```
import java.io.*;
class SumaJava {
    private native int suma(int a, int b);
    public static void main(String[] args) {
      int a=Integer.parseInt(args[0]);
      int b=Integer.parseInt(args[1]);
      SumaJava sj= new SumaJava();
      System.out.println(sj.suma(a,b));
    }
    static {
       System.loadLibrary("sumalib");
    }
 }
```

Para que éste funcione se debe crear la biblioteca de ligado dinámico. Para ello hacemos:

1. Compilamos SumaJava.java:

$ javac SumaJava.java

2. Luego creamos SumaJava.h:
   $javah SumaJava /* o $javah -jni SumaJava */
3. Ahora debemos implementar el método nativo:

```
/* en archivo sumalib.c */
#include <jni.h>
#include <stdio.h>
#include "SumaJava.h"
int suma(int, int);  /* función con implementada en C en archivo suma.o */

 JNIEXPORT jint JNICALL
 Java_SumaJava_suma(JNIEnv *env, jobject obj, jint a, jint b)
 {
   return suma(a,b);
 }
```

4. Ahora creamos la biblioteca de ligado dinámico:
   $cc -shared -I/usr/lib/jvm/java-6-sun/include -I/usr/lib/jvm/java-6-sun/include/linux -o libsumalib.so sumalib.c suma.o
5. Luego debemos asignar la variable de ambiente:
   $ export LD_LIBRARY_PATH=.
6. Ahora podemos ejecutar el programa con:
   $java SumaJava 20 34