Segundo Certamen
Tiempo 120 min. Al terminar, entregar sus respuestas vía AULA.

Usted deberá desarrollar los programas mostrados en la Figura 1.

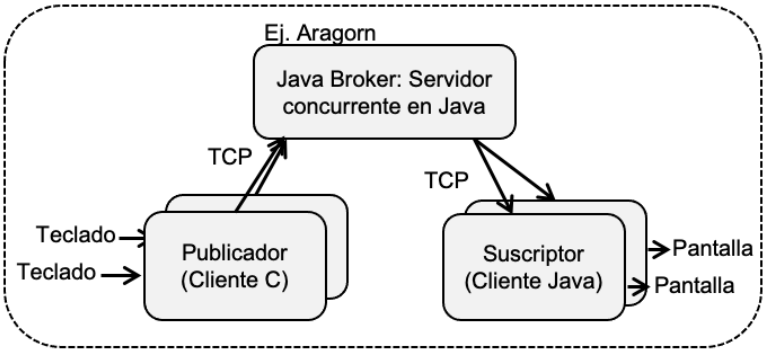

Figura 1: Programas involucrados en la situación del certamen

Idea general: Usuarios ejecutan el cliente "publicador" para enviar líneas de texto ingresadas por teclado a un servidor de distribución, el programa "Broker", el cual distribuye las líneas a cada uno de los clientes "suscriptores". Cada cliente suscriptor muestra por la salida estándar (pantalla) cada línea recibida. Un usuario podría ejecutar "publicador" en una consola y "suscriptor" en otra y así enviar y recibir mensajes a otros usuarios que hicieran lo mismo.

Pregunta 1 (30 pts): Programe en lenguaje C el cliente publicador.c.
Sintaxis: $ publicador <nombre> <servidor broker> <puerto escucha publicadores>
Descripción: publicador crea un **socket TCP** y envía string formados por la concatenación de <nombre>:<línea ingresada por teclado>; es decir, envía el argumento "nombre" seguido del separador ":" y la línea ingresada por teclado la cual debe terminar con "\n". El programa publicador termina luego de enviar el mensaje cuya línea ingresada es "bye\n".
Suba su respuesta a AULA con nombre publicador.c

Ver: http://profesores.elo.utfsm.cl/~agv/elo330/2s23/evaluaciones/C2_Solutions/publicador.c

3 Lectura de argumentos
4 Obtención de información del servicio (dirección IP, protocolo, etc.) con getaddrinfo.
4 Creación de socket
4 Conexión de socket
4 Lectura de entrada estándar
5 Concatenación de nombre como prefijo
6 Envío de mensaje y término

```c
#include <sys/types.h> //read, getaddrinfo, socket, connect
#include <sys/uio.h> //read
#include <unistd.h>   // read, close
#include <sys/socket.h> //getaddrinfo, socket, connect
#include <string.h> // memset
#include <netdb.h> // getaddrinfo
#include <stdio.h> // fprintf, printf
#include <stdlib.h> // exit

int main(int argc, char * argv[]) {
```

```
    int n, s;
    char line[1024], message[1024];
    char * userName = argv[1];
    char * brokerName = argv[2];
    char * brokerPort = argv[3];
    struct addrinfo hints, *result, *tmp;

    if (argc != 4) {
        printf("Usage: %s <nombre usuario> <servidor broker> <puerto escucha
publicadores>\n",argv[0]);
        exit(-1);
    }
    memset(&hints, 0, sizeof(hints)); /*It is important reset all to NULL*/
    hints.ai_family = AF_INET; //IPv4
    hints.ai_socktype = SOCK_STREAM; //TCP

    if ((getaddrinfo(brokerName, brokerPort, &hints, &result)) != 0) { /*Checking
status of getaddrinfo result*/
        fprintf(stderr, "getaddrinfo error\n");
        exit(EXIT_FAILURE);
    }
    for ( tmp = result; tmp != NULL; tmp = tmp->ai_next) {
      s =  socket(tmp->ai_family, tmp->ai_socktype, tmp->ai_protocol);
      if (s == -1)
        continue;
      if (connect(s, tmp->ai_addr, tmp->ai_addrlen) != 1)
        break; /*Success*/
      close(s);
    }
    freeaddrinfo(result);           /* No longer needed */
    if (tmp == NULL) { /*No address succeeded*/
           fprintf(stderr, "Could not connect\n");
           exit(EXIT_FAILURE);
    }
    /* Read from standard input, and copy the
     * data to the socket. */
    while ((n = read(STDIN_FILENO, line, sizeof(line))) > 0) {
        line[n]='\0'; // to compute strlen correctly.
        sprintf(message, "%s:%s", userName, line);
        if (send(s, message, strlen(message), 0) < 0) {
            perror("When trying to send message");
            exit(EXIT_FAILURE);
        }
        if (strcmp(line,"bye\n")==0) break;
    }
    fprintf(stderr, "Exiting ...\n"); fflush(stderr);
    close(s);
    exit(0);
}
```

Pregunta 2 (45 pts): Programe en lenguaje Java el servidor concurrente Broker.java.
Sintaxis: $ java Broker <puerto TCP de escucha publicadores> <puerto TCP escucha suscriptores>
Descripción: En “puerto TCP de escucha publicadores”, Broker atiende concurrentemente cada cliente “publicador”. A lo más habrá 5 clientes publicadores. En “puerto TCP de escucha suscriptores”, Broker atiende concurrentemente cada cliente “suscriptor”. A lo más habrá 5 clientes suscriptores.
A la llegada de un mensaje de texto desde un publicador, Broker lo distribuye a cada uno de los clientes suscriptores conectados. Broker garantiza que todos los mensajes serán enviados en el mismo orden a cada suscriptor. El programa Broker termina al presionar Control-C.
Suba su respuesta a AULA con nombre Broker.java, en él incluya otras clases posibles.

Ver: http://profesores.elo.utfsm.cl/~agv/elo330/2s23/evaluaciones/C2_Solutions/Broker.java

4 Lectura de argumentos
5 Hebra de aceptación de nuevos publicadores
5 Hebra de atención de nuevos publicadores
4 Lectura de mensajes de cada publicador
4 Envío de mensajes de cada publicador
5 Hebra de aceptación de nuevos suscriptores
8 Mecanismo para actualizar suscriptores
10 Mecanismo para envío en orden de mensajes

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

public class Broker {
   public static void main(String[] args ) {
      int publishersPort = Integer.parseInt(args[0]);
      int subscribersPort = Integer.parseInt(args[1]);
      SubscriberList subsList = new SubscriberList(); //preserve messages order
      try {
         ServerSocket pubWelcomeSock = new ServerSocket(publishersPort);
         ServerSocket subsWelcomeSock = new ServerSocket(subscribersPort);
         SubscriberWelcome sw = new SubscriberWelcome(subsWelcomeSock, subsList);
         sw.start(); // thread to accept new subscribers
         for (;;) {  // thread to accept new publishers
            Socket pubSock = pubWelcomeSock.accept( );
            System.out.println("New publisher... ");
            Thread t = new PublisherHandler(pubSock, subsList);
            t.start(); // start service for new publisher
         }
      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }
}

class SubscriberList {
   private Socket sockList[];
   private int last=0;
   public SubscriberList() {
      sockList = new Socket[5];
   }
   public synchronized void addNewSubscriber(Socket s){
      sockList[last++]=s;
   }
   public synchronized void sendMessage2All(String message) {
      for (int i=0; i<last; i++)
         try {
            sockList[i].getOutputStream().write(message.getBytes());
         } catch (IOException e0) { // this way, it detects peer leaving
            try {
               sockList[i].close();
            } catch (Exception e) {
               e.printStackTrace();
            }
            sockList[i]=sockList[--last];
            i--;
```

```
            System.out.println("A subscriber left.");
        }
    }
}

class SubscriberWelcome extends Thread {
    private ServerSocket s;
    private SubscriberList subsList;
    public SubscriberWelcome(ServerSocket welcomeSock, SubscriberList subList){
        s=welcomeSock;
        this.subsList = subList;
    }
    public void run() {
        try {
            while(true) {
                Socket incoming = s.accept( );
                System.out.println("New subscriber ...");
                subsList.addNewSubscriber(incoming);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

class PublisherHandler extends Thread {
    private Socket s;
    private SubscriberList subsList;
    public PublisherHandler(Socket sock, SubscriberList subsList) {
          s= sock;
          this.subsList = subsList;
    }
    public void run() {
        String str;
        try {
            BufferedReader in = new BufferedReader
                (new InputStreamReader(s.getInputStream()));
            boolean done = false;
            while ((str=in.readLine())!=null)
                subsList.sendMessage2All(str+'\n');
            in.close();
            System.out.println("A publisher left.");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Pregunta 3 (25 pts): Programe en lenguaje Java el cliente Suscriptor.java
Sintaxis: $ java Suscriptor <servidor Broker> <puerto escucha suscriptores>
Descripción: Suscriptor crea un **socket TCP** y espera la llegada de mensajes de texto. Suscriptor muestra por pantalla todas los mensajes que recibe desde el servidor Broker. Suscriptor termina al presionar Control-C.
Suba su respuesta a AULA con nombre Suscriptor.java, en él incluya otras clases posibles.

Ver: http://profesores.elo.utfsm.cl/~agv/elo330/2s23/evaluaciones/C2_Solutions/Suscriptor.java

5 Lectura de argumentos

5 Creación de socket conectado
10 Lectura y escritura de mensajes
5 Detección de condición de término (llega string null)

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

public class Suscriptor {
  public static void main(String[] args) throws IOException {
    String str;
    InetAddress brokerAddr =  InetAddress.getByName(args[0]);
    int brokerPort = Integer.parseInt(args[1]);
    Socket socket = new Socket(brokerAddr, brokerPort);
    try {
      BufferedReader in = new BufferedReader(new InputStreamReader(
socket.getInputStream()));
      while ((str = in.readLine()) != null)
        System.out.println(str);
    } catch (Exception e ){
      e.printStackTrace();
    } finally {
      System.out.println("Exiting ... Broker closed.");
      socket.close();
    }
  }
}
```