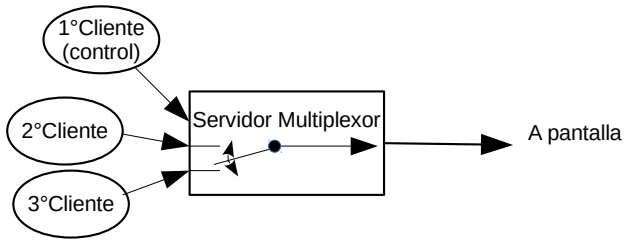
Segundo Certamen

Tiempo 110 min. Al terminar entregar sus respuestas vía AULA

Para cada pregunta, ponga su código fuente o texto en un único archivo tar , rar o zip (no archivo pdf).
Nombre sus archivos siguiendo: <P#de pregunta>_<Nombre>_<Apellido>.tar Por ejemplo: P1_Agustin_Gonzalez.tar. También pueden ser archivos .rar o .zip.

Una vez concluido el tiempo, usted tendrá 35 minutos para seguir haciendo entregas, pero se descontará dos punto por cada minuto de atraso.

1.- **Servidor TCP multiplexor de líneas:** Programe un servidor TCP que reciba conexiones de a lo más 3 clientes simultáneos. Cada cliente, posterior al primero, envía líneas de texto al servidor. El servidor muestra por pantalla las líneas provenientes de un sólo cliente y descarta las líneas enviadas por el otro cliente suministrador de líneas. El primer cliente que se conecta es de control.
El cliente cuyo texto se mostrará es definido vía comandos enviados por la primera conexión al servidor. Si el cliente de control envía la letra “n” (por next), el servidor pasa a mostrar las líneas provenientes del otro cliente. Si existe sólo un cliente, sigue atendiendo a ese único. Si el cliente de control envía una letra “s” (por stop), el servidor termina su servicio.



La figura muestra tres clientes conectados al servidor. Los tres establecen conexión a un mismo puerto. El conmutador mostrado selecciona las líneas de qué cliente son mostradas en pantalla, partiendo por el 2° cliente conectado. El cambio de fuente de líneas se produce al llegadar del comando “n” desde el 1° cliente.

1a) (44 Puntos) Programe el Servidor Multiplexor en lenguaje C y usando hebras. El servidor se ejecuta corriendo: $ multiplexor <puerto de escucha>
Suponga que todos los llamados al sistema son exitosos. La idea es que su código se concentre en la lógica significativa.
/*
Su programa debe considerar al menos lo siguiente:
Creación de hebras para atender cada clientes en main: 5
Definición de mutex y variables globales para impresión coordinada: 5
Lógica para conmutar líneas correcta (comando “n” funciona): 9
Lógica para conmutar lineas completas y no medias líneas (lectura e impresión de líneas completas): 5
Correcto uso de variables globales usando mutex: 5
Prototipo correcto para hebras: 5
Lógica para terminar ante “s”: 5
Estructura general del programa: 5
*/

```
#include <stdio.h>  /* printf */
#include <sys/socket.h>  /* socket, bind, listen */
#include <arpa/inet.h>  /* htonl */
```

```
int createTCPserverSocket(int port) {
    int s, len;
    struct sockaddr_in name;

    s = socket(AF_INET, SOCK_STREAM, 0);
    name.sin_family = AF_INET;
    name.sin_port = htons(port);
    name.sin_addr.s_addr = htonl(INADDR_ANY);
    len = sizeof(struct sockaddr_in);
    if( bind(s, (struct sockaddr *) &name, len))
      printf("bind error");
    listen(s, 5);
    return(s);
}

#include <pthread.h>
#include <stdlib.h>  /* free */
#include <unistd.h>   /* read,  close */

pthread_mutex_t mutex=PTHREAD_MUTEX_INITIALIZER;
int current;  /* index for current switch's position */
int clientSock[2];  /* sockets for both clients */

void * linesClientHandler(void * arg){  /* int arg[]  */
    int mySelf = *((int*)arg);
    char *line=NULL;
    FILE * sd;
    int n;
    size_t size=0; /* for getline to allocate memory */

    sd = fdopen(mySelf, "r");
    while (1) {
       n=getline(&line, &size, sd);
       if (n <= 0) break;
       pthread_mutex_lock(&mutex);
       if (clientSock[current]==mySelf)
          printf("%s", line);
       pthread_mutex_unlock(&mutex);
    }
    free(line);
    close(mySelf);
}
void * controlHandler(void * arg) {
    int socket = *((int*)arg);
    char c;
    while (1) {
      read (socket, &c, 1);
      if( c=='n')  {
         pthread_mutex_lock(&mutex);
         current=(current+1)%2;
         if (clientSock[current]<0)
            current=(current+1)%2;
         pthread_mutex_unlock(&mutex);
      } else  if(c=='s')  break;
    }
    close(socket);
}

int main(int argc, char *argv[])  {
    int s, n, ns, len, controlSocket;
    struct sockaddr_in name;
    pthread_t tid0, tid1, tid2;
```

```
    clientSock[0]=clientSock[1]=-1;
    current=0;
    s = createTCPserverSocket(atoi(argv[1]));
    controlSocket = accept(s, (struct sockaddr *) &name, &len);
    pthread_create(&tid0, NULL, controlHandler, &controlSocket);
    ns = accept(s, (struct sockaddr *) &name, &len);
    pthread_mutex_lock(&mutex);
    clientSock[0]=ns;
    pthread_mutex_unlock(&mutex);
    pthread_create(&tid1, NULL, linesClientHandler, &clientSock[0]);
    ns = accept(s, (struct sockaddr *) &name, &len);
    pthread_mutex_lock(&mutex);
    clientSock[1]=ns;
    pthread_mutex_unlock(&mutex);
    pthread_create(&tid2, NULL, linesClientHandler, &clientSock[1]);
    pthread_join(tid0, NULL);
    close(s);
    exit(0);
}
```

1b) (44 puntos) Programe el Servidor Multiplexor en lenguaje Java y usando hebras. El servidor se ejecuta corriendo: $ java Multiplexor <puerto de escucha>

/*
Creación de hebras para atender cada clientes en main: 5
Definición de métodos sincronizados para impresión coordinada: 8
Lógica para conmutar líneas correcta (comando “n” funciona): 11
Lógica para conmutar lineas completas y no medias líneas (lectura e impresión de líneas completas): 5
Correcta definición e inicio de clases-hebras: 5
Lógica para terminar ante “s”: 5
Estructura general de clases, definición de atributos, etc: 5
*/

```
import java.io.*;
import java.net.*;
public class Multiplexor {
   public static void main(String[] args ) {
      try {
         ServerSocket s = new ServerSocket(Integer.parseInt(args[0]));
         MySwitch mySwitch = new MySwitch();   // switch is an object
         Socket cs = s.accept( );
         ControlHandler control = new ControlHandler(cs, mySwitch);
         control.start();
         Socket incoming = s.accept( );
         LinesClientHandler linesThread = new LinesClientHandler(incoming, mySwitch);
         mySwitch.setClient1(linesThread); // let switch know a client arrives
         linesThread.start();
         incoming = s.accept( );
         linesThread = new LinesClientHandler(incoming, mySwitch);
         mySwitch.setClient2(linesThread);  // register second client with switch
         linesThread.start();
         control.join();
      } catch (Exception e) {
         e.printStackTrace();
      }
      System.exit(0);
   }
}
```

```
class MySwitch {
   synchronized public void setClient1(LinesClientHandler lch) {
     current = client1 = lch;
   }
   synchronized public void setClient2(LinesClientHandler lch) {
     client2 = lch;
   }
   synchronized public void printLine (String s, Thread thread) {
     if (current == thread)
       System.out.println(s);
   }
   synchronized public void next() {
     if ((current==client1) && (client2!=null))
        current=client2;
     else if ((current==client2) && (client1!=null))
        current=client1;
   }
   private LinesClientHandler client1, client2, current;
}
class ControlHandler extends Thread {
   public ControlHandler(Socket sock, MySwitch sw) {
      controlSock=sock;
      mySwitch = sw;
   }
   public void run() {
      try {
         BufferedReader in = new BufferedReader
            (new InputStreamReader(controlSock.getInputStream()));
         boolean done = false;
         while (!done) {
            String str = in.readLine();
            if (str == null) done = true;
            else {
               if (str.trim().equals("n"))
                  mySwitch.next();
               else if (str.trim().equals("s"))
                  done=true;
            }
         }
         controlSock.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
   private Socket controlSock;
   private MySwitch mySwitch;
}

class LinesClientHandler extends Thread {
   public LinesClientHandler(Socket i, MySwitch sw) {
      incoming = i;
      mySwitch = sw;
   }
   public void run() {
      try {
         BufferedReader in = new BufferedReader
            (new InputStreamReader(incoming.getInputStream()));
         boolean done = false;
         while (!done) {
            String str = in.readLine();
            if (str == null) done = true;
            else mySwitch.printLine(str, this);
```

```
            }
            incoming.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private Socket incoming;
    private MySwitch mySwitch;
}
```

2.- (12 puntos) Suponga que XAMPP acaba de ser actualizado y ahora usa una base de datos nueva usmDB. A usted se le pide actualizar la aplicación porque al ejecutar el programa se obtiene el mensaje “No se pudo cargar el Driver”. Muestre los pasos necesarios para obtener la información necesaria e indique los cambios necesarios al siguiente código.

```
try {
    Class.forName("org.mariadb.jdbc.Driver");
} catch( Exception e ) {
    System.out.println( "No se pudo cargar el Drive." );
    return;
}
```

1.- (6 puntos) Se debe bajar el driver o conector de usmDB para Java.
Se debe cambiar la variable CLASSPATH u opción -cp de la ejecución para que incluya el directorio donde se descargó el conector previo. (4 puntos, si pone solo 1 y 2 también se le concede puntaje total dado que es posible bajar el conector al mismo directorio donde estaba el conector previo)
2.- (6 puntos) Se debe verificar y eventualmente cambiar el path el llamado Class.forName(…). Para esto se ejecuta el comando jar con opción -tvf para identificar el path de este conector.