Contenido

[Unidad 1 Sockets de flujo 4](#_Toc36469047)

[1.3.1.a Programa de eco java 4](#_Toc36469048)

[Servidor SEcoTCPB.java 4](#_Toc36469049)

[Cliente CEcoTCPB.java 5](#_Toc36469050)

[1.3.1.b Envío de un archivo java 6](#_Toc36469051)

[Servidor SArchTCPB.java 6](#_Toc36469052)

[Cliente CArchTCPB.java 7](#_Toc36469053)

[1.3.1.c Eco en C 9](#_Toc36469054)

[MensajeError.h 9](#_Toc36469055)

[Cliente EcoTCPCliente.c 9](#_Toc36469056)

[Servidor EcoTCPServidor.c 11](#_Toc36469057)

[1.3.1.d Envío de datos primitivos 14](#_Toc36469058)

[Envia.c 14](#_Toc36469059)

[Recibe.c 17](#_Toc36469060)

[1.3.2.a Sockets de flujo no bloqueantes 19](#_Toc36469061)

[Cliente 19](#_Toc36469062)

[Servidor 22](#_Toc36469063)

[1.3.2b 24](#_Toc36469064)

[tecla.c 24](#_Toc36469065)

[1.3.2c 25](#_Toc36469066)

[servidornb.c 25](#_Toc36469067)

[clientenb.c 29](#_Toc36469068)

[1.4 Serialización 32](#_Toc36469069)

[Serialización, clase usuario 32](#_Toc36469070)

[Serialización, servidor 33](#_Toc36469071)

[Serialización, cliente 34](#_Toc36469072)

[1.4.b Extrenalizable 35](#_Toc36469073)

[Usuario.java 35](#_Toc36469074)

[DemoExternalizable.java 37](#_Toc36469075)

[2.1 Sockets de datagrama 39](#_Toc36469076)

[Servidor eco con DatagramSocket 39](#_Toc36469077)

[Cliente eco con DatagramSocket 39](#_Toc36469078)

[Servidor de datos primitivos 40](#_Toc36469079)

[Cliente de datos primitivos 41](#_Toc36469080)

[2.3.4.a Multicast 42](#_Toc36469081)

[CMulticastB.java 42](#_Toc36469082)

[SMulticastB.java 43](#_Toc36469083)

[2.3.4.b Multicast NB 44](#_Toc36469084)

[CMulticastNB.java 44](#_Toc36469085)

[SMulticastNB.java 46](#_Toc36469086)

[2.1.1. Socket multicast, bloqueantes C-java 48](#_Toc36469087)

[mcBrecibe.c 48](#_Toc36469088)

[mcBsenvia.c 52](#_Toc36469089)

[MulticastClient2.java 55](#_Toc36469090)

[MulticastServer2.java 57](#_Toc36469091)

[2.1.2 Sockets de datagrama no bloqueante 60](#_Toc36469092)

[Servidor eco UDP no bloqueante 60](#_Toc36469093)

[Cliente eco UDP no bloqueante 61](#_Toc36469094)

[4.1.a 64](#_Toc36469095)

[Hijo.c 64](#_Toc36469096)

[4.1.b 64](#_Toc36469097)

[canceHijo1.c 64](#_Toc36469098)

[4.1.c 66](#_Toc36469099)

[canceHijo2.c 66](#_Toc36469100)

[4.2.3 Serie de Taylor 68](#_Toc36469101)

[errores.h 68](#_Toc36469102)

[taylor.c 68](#_Toc36469103)

[4.2.4.a Cámbio de atributos 71](#_Toc36469104)

[atributos.c 71](#_Toc36469105)

[4.2.4.b 72](#_Toc36469106)

[Hilo\_retorno.c 72](#_Toc36469107)

[4.2.4.c Hilos y sockets 74](#_Toc36469108)

[ecoHilosServidor.c 74](#_Toc36469109)

[ecoHilosCliente.c 77](#_Toc36469110)

[4.3.3.a Candados 80](#_Toc36469111)

[Candados.c 80](#_Toc36469112)

[4.3.3.b variables de condición 81](#_Toc36469113)

[Cond.c 81](#_Toc36469114)

[4.3.3.c Semáforos 83](#_Toc36469115)

[Semaforo.c 83](#_Toc36469116)

[4.3.3.d Tuberias 84](#_Toc36469117)

[Tuberías.c 84](#_Toc36469118)

[4.3.3.e EcoHilos java 87](#_Toc36469119)

[SEcoHilos.java 87](#_Toc36469120)

[CEcoHilos.java 88](#_Toc36469121)

[4.3.3.f Mutex java 89](#_Toc36469122)

[Mutex.java 89](#_Toc36469123)

[4.3.3.g Variables de condición en java 91](#_Toc36469124)

[CondDemos.java 91](#_Toc36469125)

[Producer.java 93](#_Toc36469126)

[Consumer.java 94](#_Toc36469127)

[4.3.3.h Semáforos 95](#_Toc36469128)

[Restaurante.java 95](#_Toc36469129)

[Cliente.java 96](#_Toc36469130)

[4.3.3.i Tuberías 97](#_Toc36469131)

[Consumidor.java 97](#_Toc36469132)

[Productor.java 98](#_Toc36469133)

[4.5 Alberca de hilos 100](#_Toc36469134)

[AlbercaHilos.java 100](#_Toc36469135)

[TareaAlbercaHilos.java 100](#_Toc36469136)

[5.2 RMI 102](#_Toc36469137)

[Suma.java 102](#_Toc36469138)

[Cliente.java 102](#_Toc36469139)

[Servidor.java 103](#_Toc36469140)

# Unidad 1 Sockets de flujo

## 1.3.1.a Programa de eco java

### Servidor SEcoTCPB.java

import java.net.\*;

import java.io.\*;

public class SEcoTCPB {

public static void main(String[] args){

try{

// Se crea el socket

ServerSocket s = new ServerSocket(1234);

System.out.println("Esperando cliente ...");

// Iniciamos el ciclo infinito

for(;;){

// Tenemos un bloqueo, en el momento que llegue una conexión continua el programa

Socket cl = s.accept();

System.out.println("Conexión establecida desde "+ cl.getInetAddress()+":" + cl.getPort());

String mensaje ="Hola mundo";

PrintWriter pw = new PrintWriter(new OutputStreamWriter(cl.getOutputStream()));

// Se envía el mensaje

pw.println(mensaje);

// Se limpia le flujo

pw.flush();

pw.close();

cl.close();

}//for

}catch(Exception e){ // Manejo de excepciones

e.printStackTrace();

}//catch

}//main

}

### Cliente CEcoTCPB.java

import java.net.\*;

import java.io.\*;

public class CEcoTCPB {

public static void main(String[] args){

try{

BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));

System.out.printf("Escriba la dirección del servidor: ");

String host = br1.readLine();

System.out.printf("\n\nEscriba el puerto:");

int pto = Integer.parseInt(br1.readLine());

// Creamos el socket y nos conectamos

Socket cl = new Socket(host,pto);

BufferedReader br2 = new BufferedReader(new InputStreamReader(cl.getInputStream()));

// Nos conectamos

String mensaje = br2.readLine();

System.out.println("Recibimos un mensaje desde el servidor");

System.out.println("Mensaje:"+mensaje);

// Cerramos flujos y socket

br1.close();

br2.close();

cl.close();

}catch(Exception e){ //Manejo de excepciones

e.printStackTrace();

}

}

}

## 1.3.1.b Envío de un archivo java

### Servidor SArchTCPB.java

import java.net.\*;

import java.io.\*;

public class SArchTCPB {

public static void main(String[] args){

try{

// Creamos el socket

ServerSocket s = new ServerSocket(7000);

// Iniciamos el ciclo infinito del servidor

for(;;){

// Esperamos una conexión

Socket cl = s.accept();

System.out.println("Conexión establecida desde"+cl.getInetAddress()+":"+cl.getPort());

DataInputStream dis = new DataInputStream(cl.getInputStream());

byte[] b = new byte[1024];

String nombre = dis.readUTF();

System.out.println("Recibimos el archivo:"+nombre);

long tam = dis.readLong();

DataOutputStream dos = new DataOutputStream(new FileOutputStream(nombre));

long recibidos=0;

int n, porcentaje;

while(recibidos < tam){

n = dis.read(b);

dos.write(b,0,n);

dos.flush();

recibidos = recibidos + n;

porcentaje = (int)(recibidos\*100/tam);

System.out.print("Recibido: "+porcentaje+"%\r");

}//While

System.out.print("\n\nArchivo recibido.\n");

dos.close();

dis.close();

cl.close();

}

}catch(Exception e){

e.printStackTrace();

}//catch

}

}

### Cliente CArchTCPB.java

import javax.swing.JFileChooser;

import java.net.\*;

import java.io.\*;

public class CArchTCPB {

public static void main(String[] args){

try{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.printf("Escriba la dirección del servidor:");

String host = br.readLine();

System.out.printf("\n\nEscriba el puerto:");

int pto = Integer.parseInt(br.readLine());

Socket cl = new Socket(host, pto);

JFileChooser jf = new JFileChooser();

int r = jf.showOpenDialog(null);

if (r==JFileChooser.APPROVE\_OPTION){

File f = jf.getSelectedFile(); //Manejador

String archivo = f.getAbsolutePath(); //Dirección

String nombre = f.getName(); //Nombre

long tam = f.length(); //Tamaño

DataOutputStream dos = new DataOutputStream(cl.getOutputStream());

DataInputStream dis = new DataInputStream(new FileInputStream(archivo));

dos.writeUTF(nombre);

dos.flush();

dos.writeLong(tam);

dos.flush();

byte[] b = new byte[1024];

long enviados = 0;

int porcentaje, n;

while (enviados < tam){

n = dis.read(b);

dos.write(b,0,n);

dos.flush();

enviados = enviados+n;

porcentaje = (int)(enviados\*100/tam);

System.out.print("Enviado: "+porcentaje+"%\r");

}//While

System.out.print("\n\nArchivo enviado");

dos.close();

dis.close();

cl.close();

}//if

}catch(Exception e){

e.printStackTrace();

}

}

}

## 1.3.1.c Eco en C

### MensajeError.h

#include <stdio.h>

#include <stdlib.h>

void mensajeFinalError(const char \*mensaje){

fputs(mensaje,stderr);

fputc('\n',stderr);

exit(1);

}

void mensajeFinalSistema(const char \*mensaje){

perror(mensaje);

exit(1);

}

### Cliente EcoTCPCliente.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include "MensajeError.h"

#define TAMBUFER 2000

int main(int argc, char\*\* argv) {

if (argc < 3|| argc >4)

mensajeFinalError(

"Uso: EcoTCPCliente <Dirección del servidor> <Palabra de eco> [<Puerto>]");

char \*servIP = argv[1];

char \*cadenaEco = argv[2];

//Argumento opcional, se agrega por defecto

in\_port\_t puerto = (argc == 4) ? atoi(argv[3]) : 7;

//Crea el socket del cliente TCP

int s = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if(s < 0){

mensajeFinalError("Error de apertura del conector");

}

//Creamos la dirección del servidor de entrada

struct sockaddr\_in dirServ;

memset(&dirServ,0,sizeof(dirServ));

dirServ.sin\_family = AF\_INET;

int valRet = inet\_pton(AF\_INET,servIP,&dirServ.sin\_addr.s\_addr);

if(valRet == 0)

mensajeFinalError(

"Dirección del servidor erronea");

else if(valRet < 0)

mensajeFinalError("Error en el inet\_pton()");

dirServ.sin\_port = htons(puerto);

//Establecemos la comunicación con el servidor de eco

if(connect(s, (struct sockaddr\*) &dirServ,sizeof(dirServ))<0)

mensajeFinalError("Error en la conexión");

size\_t longCadenaEco = strlen(cadenaEco);

//Envia el mensaje al servidor

ssize\_t numBytes = send(s,cadenaEco,longCadenaEco, 0);

if(numBytes< 0)

mensajeFinalError("Fallo el envio");

else if(numBytes != longCadenaEco)

mensajeFinalError("Número de bytes enviados erroneo");

//Recibimos de vuelta la cadena desde el servidor

unsigned int totalBytesRec = 0;

while(totalBytesRec < longCadenaEco){

char bufer[TAMBUFER];

memset(bufer,0,TAMBUFER);

numBytes = recv(s,bufer,TAMBUFER, 0);

if(numBytes<0)

mensajeFinalError("Recepción fallida");

else if(numBytes==0)

mensajeFinalError(

"Conexión cerrada prematuramente");

totalBytesRec += numBytes;

printf("Recibido: %s\n",bufer);

}

close(s);

return 0;

}

### Servidor EcoTCPServidor.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include "./MensajeError.h"

#define MAXLISTA 5

#define TAMBUFER 1024

void manejadorTCPCliente(int);

int main(int argc, char \*\*argv){

if(argc != 2) //Revisamos el número de argumentos

mensajeFinalError("Uso: EcoTCPServidor [<puerto>]");

in\_port\_t prtoServ = atoi(argv[1]);

//Creamos el socket de entrada

int sockServ;

if((sockServ = socket(AF\_INET,SOCK\_STREAM,IPPROTO\_TCP)) < 0)

mensajeFinalError("Fallo la apertura del socket");

//Se construye la estructura de la dirección

struct sockaddr\_in dirServ; //Estructura para la dirección local

memset(&dirServ, 0 , sizeof(dirServ)); //Limpiamos la estructura

dirServ.sin\_family = AF\_INET; //Familia de direcciones IPv4

dirServ.sin\_addr.s\_addr = htons(INADDR\_ANY); //Cualquier interfaz de entrada​

dirServ.sin\_port = htons(prtoServ); //Número de puerto

//Se enlaza a la dirección local

if(bind(sockServ, (struct sockaddr\*)&dirServ, sizeof(dirServ))< 0)

mensajeFinalError("Error al enlazar");

//Marcamos el socket para que pueda escuchar conexiones

if(listen(sockServ, MAXLISTA) < 0)

mensajeFinalError("Error al escuchar");

for(;;){

printf("Servidor listo...\n");

struct sockaddr\_in dirCliente; //Dirección del cliente​

//Obtenemos el tamaño de la estructura

socklen\_t dirClienteTam = sizeof(dirCliente);

//Esperamos que se conecte un cliente

int sockCliente = accept(sockServ,(struct sockaddr \*)&dirCliente,&dirClienteTam);

if (sockCliente < 0)

mensajeFinalError("Fallo la conexión ");

//Se conecto un cliente

char nombreCliente[INET\_ADDRSTRLEN];

if(inet\_ntop(AF\_INET, &dirCliente.sin\_addr.s\_addr, nombreCliente,sizeof(nombreCliente)) != NULL)

printf("Cliente conectado: %s/%d\n",nombreCliente,ntohs(dirCliente.sin\_port));

else

puts("Inposible conectar el cliente");

manejadorTCPCliente(sockCliente);

} //Lazo infinito

}

void manejadorTCPCliente(int sockCliente){

char bufer[TAMBUFER];

//Recibe mensaje del cliente

ssize\_t numBytesRecibidos = recv(sockCliente, bufer, TAMBUFER, 0);

if(numBytesRecibidos < 0)

mensajeFinalError("Error en la lectura de datos recibidos");

//Envia los datos recibidos

while(numBytesRecibidos > 0){

//Eco del mensaje

ssize\_t numBytesEnviados = send(sockCliente, bufer, numBytesRecibidos, 0);

if(numBytesEnviados < 0)

mensajeFinalError("Error en el envio");

else if(numBytesEnviados == 0)

mensajeFinalError("Número de bytes enviado erroneo");

//Revisamos si hay mas datos a recibir

numBytesRecibidos = recv(sockCliente,bufer, TAMBUFER,0);

if(numBytesRecibidos < 0)

mensajeFinalError("Error en la lectura de datos recibidos");

}

close(sockCliente);

}

## 1.3.1.d Envío de datos primitivos

### Envia.c

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h> //printf, perror,fdopen

#include <string.h>

#include <netdb.h> //gethostbyname

#include <unistd.h>//close

#include <stdlib.h> //exit

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

char host[10];

int pto;

printf("Escribe la direccion del servidor:");

//gets(host);

fgets(host,sizeof(host),stdin);

printf("\nEscribe el puerto:");

scanf("%d",&pto);

fflush(stdin);

struct hostent \*dst = gethostbyname(host);

if(dst==NULL){

perror("Direccion no valida");

main(argc,argv);

}//if

struct sockaddr\_in sdir;

bzero((char \*)&sdir, sizeof(sdir));

sdir.sin\_family=AF\_INET;

sdir.sin\_port=htons(pto);

memcpy((char\*)&sdir.sin\_addr.s\_addr,dst->h\_addr, dst->h\_length);

int cd = socket(AF\_INET,SOCK\_STREAM,0);

FILE \*f = fdopen(cd,"w+");

if (connect(cd,(struct sockaddr \*)&sdir,sizeof(sdir))<0)

perror("error en funcion connect()\n");

printf("\n Conexion establecida.. enviando datos..\n");

int dato1 =5;

long dato2= 70;

float dato3=3.0f;

char \*dato4="un mensaje";

int n;

//dato1

n = write(cd,&dato1,sizeof(dato1));

if(n<0) perror("Error de escritura\n");

else if(n==0) perror("Socket cerrado error de escritura\n");

else fflush(f);

printf("Se envio el dato: %d\n",dato1);

//dato2

n= write(cd,&dato2,sizeof(dato2));

if(n<0) perror("Error de escritura\n");

else if(n==0) perror("Socket cerrado error de escritura\n");

else printf("Se envio el dato: %ld\n",dato2);

fflush(f);

//dato3

char datos[10];

sprintf(datos,"%f",dato3);

n = write(cd,datos,strlen(datos));

if(n<0)

perror("Error de escritura\n");

else if(n==0)

perror("Socket cerrado\n");

else

printf("Se envio el dato %.02f\n",dato3);

fflush(f);

//dato4

n = write(cd,dato4,strlen(dato4)+1);

if(n<0)

perror("Error de escritura\n");

else if(n==0)

perror("Socket cerrado error de escritura\n");

else

pintf("Se envio el dato: %s\n",dato4);

fflush(f);

close(cd);

fclose(f);

return 0;

}//main

### Recibe.c

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <string.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <unistd.h>

#include <arpa/inet.h>

int main(int agrc, char\* argv[]){

struct sockaddr\_in sdir,cdir;

bzero((char \*)&sdir, sizeof(sdir));

sdir.sin\_family=AF\_INET;

sdir.sin\_port=htons(9876);

sdir.sin\_addr.s\_addr=htonl(INADDR\_ANY);

socklen\_t ctam = sizeof(cdir);

int sd,cd,v=1,op;

if((sd=socket(AF\_INET,SOCK\_STREAM,0))<0)

perror("Error en la funcion socket()\n");

op=setsockopt(sd,SOL\_SOCKET,SO\_REUSEADDR,&v,sizeof(v));

if(op<0) perror("Error en la opcion de socket\n");

if(bind(sd,(struct sockaddr \*)&sdir,sizeof(sdir))<0){

close(sd);

perror("El puerto ya esta en uso\n");

}

printf("Servicio iniciado..");

listen(sd,5);

for(;;){

if((cd=accept(sd,(struct sockaddr \*)&cdir,&ctam))<0){

perror("Error en funcion accept()\n");

continue;

}//if

printf("\nCliente conectado desde ->%s:%d\n"

"Recibiendo datos...\n",

inet\_ntoa(cdir.sin\_addr),ntohs(cdir.sin\_port));

int n,dato1;

long dato2;

float dato3;

char dato4[50];

bzero(dato4,sizeof(dato4));

//dato1

n = read(cd,&dato1, sizeof(dato1));

printf("dato1->%d\n",dato1);

//dato2

n= read(cd,&dato2, sizeof(dato2));

printf("dato2->%ld\n",dato2);

//dato3

char datos[20];

n= read(cd,datos, sizeof(datos));

dato3= atof(datos);

printf("dato3->%.02f\n",dato3);

//dato4

n= read(cd,dato4, sizeof(dato4));

printf("dato4->%s\n",dato4);

close(cd);

}//for

close(sd);

return 0;

}//main

## 1.3.2.a Sockets de flujo no bloqueantes

### Cliente

import java.nio.channels.\*;

import java.io.\*;

import java.net.\*;

import java.nio.ByteBuffer;

import java.util.Iterator;

public class CEcoTCPNB {

public static void main(String[] args){

try{

String dir="127.0.0.1";

int pto = 9999;

ByteBuffer b1=null, b2=null;

InetSocketAddress dst = new InetSocketAddress(dir,pto);

SocketChannel cl = SocketChannel.open();

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

cl.configureBlocking(false);

Selector sel = Selector.open();

cl.register(sel, SelectionKey.OP\_CONNECT);

cl.connect(dst);

while(true){

sel.select();

Iterator<SelectionKey>it = sel.selectedKeys().iterator();

while(it.hasNext()){

SelectionKey k = (SelectionKey)it.next();

it.remove();

if(k.isConnectable()){

SocketChannel ch = (SocketChannel)k.channel();

if(ch.isConnectionPending()){

System.out.println("Estableciendo conexion con el servidor... espere..");

try{

ch.finishConnect();

}catch(Exception e){

e.printStackTrace();

}//catch

System.out.println("Conexion establecida...\nEscribe texto <Enter> para enviar, SALIR para terminar:");

}//if

ch.register(sel, SelectionKey.OP\_READ|SelectionKey.OP\_WRITE);

continue;

}//if

if(k.isReadable()){

SocketChannel ch = (SocketChannel)k.channel();

b1 = ByteBuffer.allocate(2000);

b1.clear();

int n = ch.read(b1);

b1.flip();

String eco = new String(b1.array(),0,n);

System.out.println("Eco de "+n+" bytes recibido: "+eco);

k.interestOps(SelectionKey.OP\_WRITE);

continue;

} else if(k.isWritable()){

SocketChannel ch = (SocketChannel)k.channel();

String datos="";

datos=br.readLine();

if (datos.equalsIgnoreCase("SALIR")){

System.out.println("Termina aplicacion...");

byte[]mm = "SALIR".getBytes();

b2 = ByteBuffer.wrap(mm);

ch.write(b2);

k.interestOps(SelectionKey.OP\_READ);

k.cancel();

ch.close();

System.exit(0);

}//if

byte[]mm = datos.getBytes();

System.out.println("Enviando eco de "+mm.length+" bytes..");

b2 = ByteBuffer.wrap(mm);

ch.write(b2);

k.interestOps(SelectionKey.OP\_READ);

continue;

} //if

}//while

}//while

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

### Servidor

import java.io.\*;

import java.net.\*;

import java.nio.ByteBuffer;

import java.nio.channels.\*;

import java.util.Iterator;

public class SEcoTCPNB {

public static void main(String[] args){

try{

String EECO="";

int pto=9999;

ServerSocketChannel s = ServerSocketChannel.open();

s.configureBlocking(false);

s.socket().bind(new InetSocketAddress(pto));

System.out.println("Esperando clientes...");

Selector sel = Selector.open();

s.register(sel, SelectionKey.OP\_ACCEPT);

while(true){

sel.select();

Iterator<SelectionKey>it = sel.selectedKeys().iterator();

while(it.hasNext()){

SelectionKey k = (SelectionKey)it.next();

it.remove();

if(k.isAcceptable()){

SocketChannel cl = s.accept();

System.out.println("Cliente conectado desde + cl.socket().getInetAddress() + ":"+

cl.socket().getPort());

cl.configureBlocking(false);

cl.register(sel,SelectionKey.OP\_READ|SelectionKey.OP\_WRITE);

continue;

}//if

if(k.isReadable()){

try{

SocketChannel ch = (SocketChannel)k.channel();

ByteBuffer b = ByteBuffer.allocate(2000);

b.clear();

int n=0;

String msj="";

n=ch.read(b);

b.flip();

if(n>0)

msj = new String(b.array(),0,n);

System.out.println("Mensaje de "+n+" bytes recibido: "+msj);

if (msj.equalsIgnoreCase("SALIR")){

k.interestOps(SelectionKey.OP\_WRITE);

ch.close();

// k.cancel();

}else{

EECO="ECO->"+msj;

k.interestOps(SelectionKey.OP\_WRITE);

}//else

}catch(IOException io){}

continue;

}else if(k.isWritable()){

try{

SocketChannel ch = (SocketChannel)k.channel();

ByteBuffer bb = ByteBuffer.wrap(EECO.getBytes());

ch.write(bb);

System.out.println("Mensaje de "+EECO.length() +" bytes enviado: "+EECO);

EECO="";

}catch(IOException io){}

k.interestOps(SelectionKey.OP\_READ);

continue;

}//if

}//while

}//while

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

## 1.3.2b

### tecla.c

#include <stdio.h>

#include <sys/time.h>

#include <sys/types.h>

#include <unistd.h>

#define STDIN 0 // descriptor de la entrada estándar

int main(void){

fd\_set readfds;

struct timeval tv;

tv.tv\_sec = 2;

tv.tv\_usec = 500000;

FD\_ZERO(&readfds);

FD\_SET(STDIN, &readfds);

select(STDIN+1, &readfds, NULL, NULL, &tv); // no nos preocupemos de writefds y exceptfds:

if (FD\_ISSET(STDIN, &readfds))

printf("Se presiono una tecla\n");

else

printf("Expiro el tiempo\n");

return 0;

}

## 1.3.2c

### servidornb.c

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/wait.h>

#include <sys/time.h>

#include <sys/types.h>

#define MAXBUF 1024

typedef struct CLIENTE {

int fd;

struct sockaddr\_in addr;

}CLIENTE;

int main(int argc, char\*\* argv){

int i,n,maxi = -1;

int nlisto;

int slisten,sockfd,maxfd=-1,conectafd;

unsigned int puerto,lisnum;

struct sockaddr\_in my\_addr,addr;

struct timeval tv;

  socklen\_t len;

fd\_set rset,allset;

  char buf[MAXBUF + 1];

CLIENTE cliente[FD\_SETSIZE];

if(argv[1]) puerto = atoi(argv[1]);

else puerto = 1234;

if(argv[2]) lisnum = atoi(argv[2]);

else lisnum = FD\_SETSIZE; //Tamaño máximo de elementos de un conjunto fd\_set

if((slisten = socket(AF\_INET,SOCK\_STREAM,0)) == -1){

perror("Error en socket()\n");

exit(1);

}

int flags = fcntl(slisten, F\_GETFL, 0);

fcntl(slisten, F\_SETFL, flags | O\_NONBLOCK);

bzero(&my\_addr,sizeof(my\_addr));

my\_addr.sin\_family = AF\_INET;

my\_addr.sin\_port = htons(puerto);

my\_addr.sin\_addr.s\_addr = INADDR\_ANY;

if(bind(slisten, (struct sockaddr \*)&my\_addr, sizeof(my\_addr)) == -1) {

perror("Error en bind()\n");

exit(1);

}

if (listen(slisten, lisnum) == -1) {

perror("Error en listen()\n");

exit(1);

}

for(i=0;i<FD\_SETSIZE;i++) cliente[i].fd = -1;

FD\_ZERO(&allset);

FD\_SET(slisten, &allset);

maxfd = slisten;

printf("Servidor listo para recibir un máximo de %d conexiones...\n",FD\_SETSIZE);

while (1){

rset = allset;

tv.tv\_sec = 1;

tv.tv\_usec = 0;

nlisto = select(maxfd + 1, &rset, NULL, NULL, &tv);

if(nlisto == 0) continue;

else if(nlisto < 0){

printf("Error en select()\n");

break;

}else{

if(FD\_ISSET(slisten,&rset)){ //Nuevas conexiones

len = sizeof(struct sockaddr);

if((conectafd = accept(slisten,(struct sockaddr\*)&addr,&len)) == -1){

perror("Error en accept()\n");

continue;

}

for(i=0;i<FD\_SETSIZE;i++){

if(cliente[i].fd < 0){

cliente[i].fd = conectafd;

cliente[i].addr = addr;

printf("Se establecio una conexión desde %s.\n",

inet\_ntoa(cliente[i].addr.sin\_addr));

break;

}

}

if(i == FD\_SETSIZE)

printf("Demasiadas conexiones\n");

FD\_SET(conectafd,&allset);

if(conectafd > maxfd)

maxfd = conectafd;

if(i > maxi)

maxi = i;

}else{

for(i=0;i<=maxi;i++){

if((sockfd = cliente[i].fd)<0) continue;

if(FD\_ISSET(sockfd,&rset)){

bzero(buf,MAXBUF + 1);

if((n = recv(sockfd,buf,MAXBUF,0)) > 0){

printf("Datos recibidos:%s\n desde %s\n",buf,inet\_ntoa(cliente[i].addr.sin\_addr));

}else{

printf("Desconectado por el cliente\n");

close(sockfd);

FD\_CLR(sockfd,&allset);

cliente[i].fd = -1;

}

}

}

}

}

}

close(slisten);

}

### clientenb.c

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <sys/time.h>

#include <sys/types.h>

#define MAXBUF 1024

#include <fcntl.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <sys/socket.h>

#include <resolv.h>

#include <stdlib.h>

int main(int argc, char \*\*argv){

int sockfd, lon;

struct sockaddr\_in dest;

char buf[MAXBUF + 1];

fd\_set rfds;

struct timeval tv;

int retval, maxfd = -1;

if (argc != 3) {

printf("Uso: %s IP Puerto\n",argv[0]);

exit(0);

}

if ((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("Error en socket()\n");

exit(errno);

}

int val=1;

int op = setsockopt(sockfd, SOL\_SOCKET, SO\_REUSEADDR, &val, sizeof(val));

if(op<0){

perror("No se modificó opción de socket\n");

}

bzero(&dest, sizeof(dest));

dest.sin\_family = AF\_INET;

dest.sin\_port = htons(atoi(argv[2]));

if (inet\_aton(argv[1], (struct in\_addr \*) &dest.sin\_addr.s\_addr) == 0) {

perror(argv[1]);

exit(errno);

}

if(connect(sockfd, (struct sockaddr \*) &dest, sizeof(dest)) != 0) {

perror("Error en connect()\n");

exit(errno);

}

printf("Conectado al servidor...\n");

while (1){

FD\_ZERO(&rfds);

FD\_SET(0, &rfds);

maxfd = 0;

FD\_SET(sockfd, &rfds);

if (sockfd > maxfd)

maxfd = sockfd;

tv.tv\_sec = 1;

tv.tv\_usec = 0;

retval = select(maxfd + 1, &rfds, NULL, NULL, &tv);

if (retval == -1){

printf("Error en select() %s\n", strerror(errno));

break;

} else if (retval == 0) {

continue;

} else {

if(FD\_ISSET(0, &rfds)){

bzero(buf, MAXBUF + 1);

fgets(buf, MAXBUF, stdin);

if (!strncasecmp(buf, "salir", 4)){

printf("Solicitud de terminación de chat\n");

break;

}

lon = send(sockfd, buf, strlen(buf) - 1, 0);

if (lon > 0)

printf("msg:%s envio realizado, total de bytes: %d!\n", buf, lon);

else {

printf("msg:'%s error en el envio del mensaje\n", buf);

break;

}

} else if (FD\_ISSET(sockfd, &rfds)){

bzero(buf, MAXBUF + 1);

lon = recv(sockfd, buf, MAXBUF, 0);

if (lon > 0)

printf("recv:'%s, total: %d \n", buf, lon);

else{

if (lon < 0)

printf("Error en recv(); errno:%d,error msg: '%s'\n", errno, strerror(errno));

else

printf("Terminación del chat\n");

break;

}

}

}

}

close(sockfd);

return 0;

}

## 1.4 Serialización

### Serialización, clase usuario

import java.io.Serializable;

public class Usuario implements Serializable {

String nombre;

String apaterno;

String amaterno;

transient String pwd;

int edad;

public Usuario(String nombre, String apaterno, String amaterno, String pwd, int edad) {

this.nombre = nombre;

this.apaterno = apaterno;

this.amaterno = amaterno;

this.pwd = pwd;

this.edad = edad;

}

public String getNombre() {

return nombre;

}

public String getApaterno() {

return apaterno;

}

public String getAmaterno() {

return amaterno;

}

public String getPwd() {

return pwd;

}

public int getEdad() {

return edad;

}

}

### Serialización, servidor

import java.net.\*;

import java.io.\*;

public class Servidor {

public static void main(String[] args){

ObjectOutputStream oos = null;

ObjectInputStream ois= null;

try{

ServerSocket s = new ServerSocket(9999);

System.out.println("Servicio iniciado... Esperando cliente");

for(;;){

Socket cl= s.accept();

System.out.println("Cliente conectado desde "+cl.getInetAddress()+":"+cl.getPort());

oos= new ObjectOutputStream(cl.getOutputStream());

ois = new ObjectInputStream(cl.getInputStream());

Usuario u = (Usuario)ois.readObject();

System.out.println("Objeto recibido.. Extrayendo informacion");

System.out.println("Nombre = "+u.getNombre());

System.out.println("Apellido Paterno = "+u.getApaterno());

System.out.println("Apellido Materno = "+u.getAmaterno());

System.out.println("Password = "+u.getPwd());

System.out.println("Edad = "+u.getEdad());

System.out.println("Devolviendo objeto...");

oos.writeObject(u);

oos.flush();

}//for

} catch(Exception e){

e.printStackTrace();

}//catch

}//main

}//class

### Serialización, cliente

import java.net.\*;

import java.io.\*;

public class Cliente {

public static void main(String[] args){

String host= "127.0.0.1";

int pto = 9999;

ObjectOutputStream oos = null;

ObjectInputStream ois = null;

try{

Socket cl = new Socket(host,pto);

System.out.println("Conexión establecida...");

oos= new ObjectOutputStream(cl.getOutputStream());

ois = new ObjectInputStream(cl.getInputStream());

Usuario u = new Usuario("Pepito", "Perez","Juarez","12345",20);

System.out.println("Enviando objeto");

oos.writeObject(u);

oos.flush();

System.out.println("Preparado para recibir respuesta...");

Usuario u2 = (Usuario)ois.readObject();

System.out.println("Objeto recibido.. Extrayendo informacion");

System.out.println("Nombre = "+u2.getNombre());

System.out.println("Apellido Paterno = "+u2.getApaterno());

System.out.println("Apellido Materno = "+u2.getAmaterno());

System.out.println("Password = "+u2.getPwd());

System.out.println("Edad = "+u2.getEdad());

}catch(Exception e){

System.err.println(e);

}//catch

}//main

}//class

## 1.4.b Extrenalizable

### Usuario.java

import java.io.\*;

import java.util.\*;

class Usuario implements Externalizable{

private String usuario;

private String password;

public Usuario(){

System.out.println("Creando usuario vacio");

}

Usuario(String u, String p){

System.out.println("Creando usuario ("+u+","+p+")");

usuario = u;

password = p;

}

public void writeExternal(ObjectOutput out) throws IOException{

System.out.println("Usuario.writeExternal");

//Explicitamente indicamos cuales son los atributos a almacenar

out.writeObject(usuario);

}

public void readExternal(ObjectInput in)

throws IOException, ClassNotFoundException{

System.out.println("Usuario.readExternal");

//Explicitamente indicamos cuales son los atributos a recuperar

usuario = (String)in.readObject();

}

public void muestraUsuario(){

String cad = "Usuario: "+usuario+" Password: ";

if (password == null)

cad = cad + "No disponible";

else

cad = cad + password;

System.out.println(cad);

}

}

class ListaUsuarios implements Serializable{

private LinkedList lista = new LinkedList();

int valor;

ListaUsuarios(String[] usuarios, String[] passwords){

for(int i =0; i < usuarios.length; i++)

lista.add(new Usuario(usuarios[i],passwords[i]));

}

public void muestraUsuario(){

ListIterator li = lista.listIterator();

Usuario u;

while(li.hasNext()){

u = (Usuario) li.next();

u.muestraUsuario();

}

}

}

### DemoExternalizable.java

import java.io.\*;

import java.util.\*;

class DemoExternalizable{

public static void main(String[] args)

throws IOException, ClassNotFoundException{

System.out.println("Creando el objeto");

String[] usuarios = {"A","B","C"};

String[] passwords ={"1","2","3"};

ListaUsuarios lp = new ListaUsuarios(usuarios, passwords);

ObjectOutputStream o =

new ObjectOutputStream(

new FileOutputStream("objetos.out"));

o.writeObject(lp);

o.close();

System.out.println("\nRecuperando objeto");

ObjectInputStream in =

new ObjectInputStream(

new FileInputStream("objetos.out"));

lp = (ListaUsuarios) in.readObject();

lp.muestraUsuario();

}

}

## 2.1 Sockets de datagrama

### Servidor eco con DatagramSocket

import java.net.\*;

import java.io.\*;

public class SEcoUDPB {

public static void main(String[] args){

try{

DatagramSocket s = new DatagramSocket(2000);

System.out.println("Servidor iniciado, esperando cliente");

for(;;){

DatagramPacket p = new DatagramPacket(new byte[2000],2000);

s.receive(p);

System.out.println("Datagrama recibido desde"+p.getAddress()+":"+p.getPort());

String msj = new String(p.getData(),0,p.getLength());

System.out.println("Con el mensaje:"+ msj);

}//for

//s.close()

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

### Cliente eco con DatagramSocket

import java.net.\*;

import java.io.\*;

public class CEcoUDPB {

public static void main(String[] args){

try{

DatagramSocket cl = new DatagramSocket();

System.out.print("Cliente iniciado, escriba un mensaje de saludo:");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String mensaje = br.readLine();

byte[] b = mensaje.getBytes();

String dst = "127.0.0.1";

int pto = 2000;

DatagramPacket p = new DatagramPacket(b,b.length,InetAddress.getByName(dst),pto);

cl.send(p);

cl.close();

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

### Servidor de datos primitivos

import java.net.\*;

import java.io.\*;

public class SPrimUDPB {

public static void main(String[] args){

try{

DatagramSocket s = new DatagramSocket(2000);

System.out.println("Servidor iniciado, esperando cliente");

for(;;){

DatagramPacket p = new DatagramPacket(new byte[2000],2000);

s.receive(p);

System.out.println("Datagrama recibido desde"+p.getAddress()+":"+p.getPort());

DataInputStream dis = new DataInputStream(new ByteArrayInputStream(p.getData()));

int x = dis.readInt();

float f = dis.readFloat();

long z = dis.readLong();

System.out.println("\n\nEntero:"+ x + " Flotante:"+f+" Entero largo:"+z);

}//for

//s.close()

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

### Cliente de datos primitivos

import java.net.\*;

import java.io.\*;

public class CPrimUDPB {

public static void main(String[] args){

try{

int pto = 2000;

InetAddress dst = InetAddress.getByName("127.0.0.1");

DatagramSocket cl = new DatagramSocket();

ByteArrayOutputStream baos = new ByteArrayOutputStream();

DataOutputStream dos = new DataOutputStream(baos);

dos.writeInt(4);

dos.flush();

dos.writeFloat(4.1f);

dos.flush();

dos.writeLong(72);

dos.flush();

byte[] b = baos.toByteArray();

DatagramPacket p = new DatagramPacket(b,b.length,dst,pto);

cl.send(p);

cl.close();

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

## 2.3.4.a Multicast

### CMulticastB.java

import java.net.\*;

public class CMulticastB {

public static void main(String[] args ){

InetAddress gpo=null;

try{

MulticastSocket cl= new MulticastSocket(9999);

System.out.println("Cliente escuchando puerto "+ cl.getLocalPort());

cl.setReuseAddress(true);

try{

gpo = InetAddress.getByName("228.1.1.1");

}catch(UnknownHostException u){

System.err.println("Direccion no valida");

}//catch

cl.joinGroup(gpo);

System.out.println("Unido al grupo");

for(;;){

DatagramPacket p = new DatagramPacket(new byte[10],10);

cl.receive(p);

String msj = new String(p.getData());

System.out.println("Datagrama recibido.."+msj);

System.out.println("Servidor descubierto:" + p.getAddress()+" puerto:"+p.getPort());

}//for

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

### SMulticastB.java

import java.net.\*;

public class SMulticastB {

public static void main(String[] args ){

InetAddress gpo;

try{

MulticastSocket s= new MulticastSocket(9876);

s.setReuseAddress(true);

s.setTimeToLive(1);

String msj ="hola";

byte[] b = msj.getBytes();

gpo = InetAddress.getByName("228.1.1.1");

s.joinGroup(gpo);

for(;;){

DatagramPacket p = new DatagramPacket(b,b.length,gpo,9999);

s.send(p);

System.out.println("Enviando mensaje "+msj+ " con un TTL= "+s.getTimeToLive());

try{

Thread.sleep(3000);

}catch(InterruptedException ie){

ie.printStackTrace();

}

}//for

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

## 2.3.4.b Multicast NB

### CMulticastNB.java

import java.net.\*;

import java.nio.\*;

import java.nio.channels.\*;

import java.util.Collections;

import java.util.Enumeration;

import java.util.Iterator;

public class CMulticastNB {

static void displayInterfaceInformation(NetworkInterface netint) throws SocketException {

System.out.printf("Interfaz: %s\n", netint.getDisplayName());

System.out.printf("Nombre: %s\n", netint.getName());

Enumeration<InetAddress> inetAddresses = netint.getInetAddresses();

for (InetAddress inetAddress : Collections.list(inetAddresses)) {

System.out.printf("InetAddress: %s\n", inetAddress);

}

System.out.printf("\n");

}

public static void main(String[] args){

int pto=2000;

String hhost="230.0.0.1";

SocketAddress remote=null;

try{

try{

remote = new InetSocketAddress(hhost, pto);

}catch(Exception e){

e.printStackTrace();

}//catch

Enumeration<NetworkInterface> nets = NetworkInterface.getNetworkInterfaces();

for (NetworkInterface netint : Collections.list(nets))

displayInterfaceInformation(netint);

NetworkInterface ni = NetworkInterface.getByName("eth3");

DatagramChannel cl =DatagramChannel.open(StandardProtocolFamily.INET);

cl.setOption(StandardSocketOptions.SO\_REUSEADDR, true);

cl.setOption(StandardSocketOptions.IP\_MULTICAST\_IF, ni);

cl.configureBlocking(false);

Selector sel = Selector.open();

cl.register(sel, SelectionKey.OP\_READ|SelectionKey.OP\_WRITE);

InetAddress group = InetAddress.getByName("230.0.0.1");

cl.join(group, ni);

ByteBuffer b = ByteBuffer.allocate(4);

int n=0;

while(n<100){

sel.select();

Iterator<SelectionKey>it = sel.selectedKeys().iterator();

while(it.hasNext()){

SelectionKey k = (SelectionKey)it.next();

it.remove();

if(k.isWritable()){

DatagramChannel ch = (DatagramChannel)k.channel();

b.clear();

b.putInt(n++);

b.flip();

ch.send(b, remote);

continue;

}

}//while

}//while

cl.close();

System.out.println("Termina envio de datagramas");

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

### SMulticastNB.java

import java.net.\*;

import java.nio.\*;

import java.nio.channels.\*;

import java.util.Collections;

import java.util.Enumeration;

import java.util.Iterator;

public class SMulticastNB {

static void displayInterfaceInformation(NetworkInterface netint) throws SocketException {

System.out.printf("Interfaz: %s\n", netint.getDisplayName());

System.out.printf("Nombre: %s\n", netint.getName());

Enumeration<InetAddress> inetAddresses = netint.getInetAddresses();

for (InetAddress inetAddress : Collections.list(inetAddresses))

System.out.printf("InetAddress: %s\n", inetAddress);

System.out.printf("\n");

}

public static void main(String[] args){

try{

int pto=2000;

Enumeration<NetworkInterface> nets = NetworkInterface.getNetworkInterfaces();

for (NetworkInterface netint : Collections.list(nets))

displayInterfaceInformation(netint);

NetworkInterface ni = NetworkInterface.getByName("eth3");

InetSocketAddress dir = new InetSocketAddress(pto);

DatagramChannel s = DatagramChannel.open(StandardProtocolFamily.INET);

s.setOption(StandardSocketOptions.SO\_REUSEADDR, true);

s.setOption(StandardSocketOptions.IP\_MULTICAST\_IF, ni);

InetAddress group = InetAddress.getByName("230.0.0.1");

s.join(group, ni);

s.configureBlocking(false);

s.socket().bind(dir);

Selector sel = Selector.open();

s.register(sel, SelectionKey.OP\_READ);

ByteBuffer b = ByteBuffer.allocate(4);

System.out.println("Servidor listo.. Esperando datagramas...");

while(true){

sel.select();

Iterator<SelectionKey>it = sel.selectedKeys().iterator();

while(it.hasNext()){

SelectionKey k = (SelectionKey)it.next();

it.remove();

if(k.isReadable()){

DatagramChannel ch = (DatagramChannel)k.channel();

b.clear();

SocketAddress emisor = ch.receive(b);

b.flip();

InetSocketAddress d = (InetSocketAddress)emisor;

System.out.println("Datagrama recibido desde "+ d.getAddress()+":"+d.getPort());

System.out.println("Dato: "+b.getInt());

continue;

}

}//while

}//while

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

## 2.1.1. Socket multicast, bloqueantes C-java

### recibe.c

#include <sys/types.h> /\* Para definición de tipos \*/

#include <sys/socket.h> /\* Para llamadas a las APIs de socket \*/

#include <netinet/in.h> /\* Estructuras de datos \*/

#include <arpa/inet.h> /\* Para sockaddr\_in \*/

#include <stdio.h> /\* Para printf() y fprintf() \*/

#include <stdlib.h> /\* Para atoi() \*/

#include <string.h> /\* Para strlen() \*/

#include <unistd.h> /\* Para close() \*/

#define MAX\_LEN 1024 /\* Tamaño máximo de lectura \*/

#define MIN\_PORT 1024 /\* Primer puerto asignable\*/

#define MAX\_PORT 65535 /\* Último puerto asignable \*/

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

int sock; /\* Descriptor de socket \*/

int flag\_on = 1; /\* Banderas de opción del socket \*/

struct sockaddr\_in mc\_addr; /\* Estructura de la dirección del socket \*/

char recv\_str[MAX\_LEN+1]; /\* Bufer de lectura\*/

int recv\_len; /\* Longitud de la cadena recibida \*/

struct ip\_mreq mc\_req; /\* Estructura de la solicitud multicast \*/

char\* mc\_addr\_str; /\* Dirección IP multicast \*/

unsigned short mc\_port; /\* Puerto multicast \*/

struct sockaddr\_in from\_addr; /\* Paquete origen \*/

unsigned int from\_len; /\* Longitd dirección fuente \*/

/\* Validación de parámetros \*/

if (argc != 3) {

fprintf(stderr,

"Uso: %s <Multicast IP> <Multicast Puerto>\n",

argv[0]);

exit(1);

}

mc\_addr\_str = argv[1]; /\* arg 1: dirección ip de multicast\*/

mc\_port = atoi(argv[2]); /\* arg 2: número de puerto multicast \*/

/\* Validar el número de puerto \*/

if ((mc\_port < MIN\_PORT) || (mc\_port > MAX\_PORT)) {

fprintf(stderr, "Número de puerto inválido %d.\n",mc\_port);

fprintf(stderr, "El rángo válido esta entre %d y %d.\n",MIN\_PORT, MAX\_PORT);

exit(1);

}

/\* Se crea un socket para conectarse a un canal multicast \*/

if ((sock = socket(PF\_INET, SOCK\_DGRAM, IPPROTO\_UDP)) < 0) {

perror("Error en socket().");

exit(1);

}

/\* Permite la reutilización del socket \*/

if ((setsockopt(sock, SOL\_SOCKET, SO\_REUSEADDR, &flag\_on,sizeof(flag\_on))) < 0) {

perror("Error en setsockopt()");

exit(1);

}

/\* Se construye una estructura de dirección sock\_addr \*/

memset(&mc\_addr, 0, sizeof(mc\_addr));

mc\_addr.sin\_family = AF\_INET;

mc\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

mc\_addr.sin\_port = htons(mc\_port);

/\* Liga la dirección con el socket \*/

if ((bind(sock, (struct sockaddr \*) &mc\_addr,

sizeof(mc\_addr))) < 0) {

perror("Error en bind()");

exit(1);

}

/\* Estructura para unirse al grupo \*/

mc\_req.imr\_multiaddr.s\_addr = inet\_addr(mc\_addr\_str);

mc\_req.imr\_interface.s\_addr = htonl(INADDR\_ANY);

/\* Se envia el mensaje de unirse al grupo via setsockopt \*/

if ((setsockopt(sock, IPPROTO\_IP, IP\_ADD\_MEMBERSHIP, (void\*) &mc\_req, sizeof(mc\_req))) < 0) {

perror("Error en setsockopt()");

exit(1);

}

for (;;) { // lazo infinito

/\* Se limpia el buffer y la estructura de lectura \*/

memset(recv\_str, 0, sizeof(recv\_str));

from\_len = sizeof(from\_addr);

memset(&from\_addr, 0, from\_len);

/\* Bloqueo para la recepción de paquetes \*/

if ((recv\_len = recvfrom(sock, recv\_str, MAX\_LEN, 0, (struct sockaddr\*)&from\_addr, &from\_len)) < 0) {

perror("Error en recvfrom()");

exit(1);

}

/\* Inprimimos lo que recibimos \*/

printf("\nSe recibieron %d bytes desde %s: ", recv\_len, inet\_ntoa(from\_addr.sin\_addr));

printf("%s", recv\_str);

}

/\* Envia un mensaje para dejar el grupo via setsockopt \*/

if ((setsockopt(sock, IPPROTO\_IP, IP\_DROP\_MEMBERSHIP, (void\*) &mc\_req, sizeof(mc\_req))) < 0) {

perror("Error en setsockopt()");

exit(1);

}

close(sock);

}

### envia.c

#include <sys/types.h> /\* for type definitions \*/

#include <sys/socket.h> /\* for socket API function calls \*/

#include <netinet/in.h> /\* for address structs \*/

#include <arpa/inet.h> /\* for sockaddr\_in \*/

#include <stdio.h> /\* for printf() \*/

#include <stdlib.h> /\* for atoi() \*/

#include <string.h> /\* for strlen() \*/

#include <unistd.h> /\* for close() \*/

#define MAX\_LEN 1024 /\* maximum string size to send \*/

#define MIN\_PORT 1024 /\* minimum port allowed \*/

#define MAX\_PORT 65535 /\* maximum port allowed \*/

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

int sock; /\* socket descriptor \*/

char send\_str[MAX\_LEN]; /\* string to send \*/

struct sockaddr\_in mc\_addr; /\* socket address structure \*/

unsigned int send\_len; /\* length of string to send \*/

char\* mc\_addr\_str; /\* multicast IP address \*/

unsigned short mc\_port; /\* multicast port \*/

unsigned char mc\_ttl=1; /\* time to live (hop count) \*/

/\* validate number of arguments \*/

if (argc != 3) {

fprintf(stderr,

"Usage: %s <Multicast IP> <Multicast Port>\n",

argv[0]);

exit(1);

}

mc\_addr\_str = argv[1]; /\* arg 1: multicast IP address \*/

mc\_port = atoi(argv[2]); /\* arg 2: multicast port number \*/

/\* validate the port range \*/

if ((mc\_port < MIN\_PORT) || (mc\_port > MAX\_PORT)) {

fprintf(stderr, "Invalid port number argument %d.\n",

mc\_port);

fprintf(stderr, "Valid range is between %d and %d.\n",

MIN\_PORT, MAX\_PORT);

exit(1);

}

/\* create a socket for sending to the multicast address \*/

if ((sock = socket(PF\_INET, SOCK\_DGRAM, IPPROTO\_UDP)) < 0) {

perror("socket() failed");

exit(1);

}

/\* set the TTL (time to live/hop count) for the send \*/

if ((setsockopt(sock, IPPROTO\_IP, IP\_MULTICAST\_TTL,

(void\*) &mc\_ttl, sizeof(mc\_ttl))) < 0) {

perror("setsockopt() failed");

exit(1);

}

/\* construct a multicast address structure \*/

memset(&mc\_addr, 0, sizeof(mc\_addr));

mc\_addr.sin\_family = AF\_INET;

mc\_addr.sin\_addr.s\_addr = inet\_addr(mc\_addr\_str);

mc\_addr.sin\_port = htons(mc\_port);

printf("Begin typing (return to send, ctrl-C to quit):\n");

/\* clear send buffer \*/

memset(send\_str, 0, sizeof(send\_str));

while (fgets(send\_str, MAX\_LEN, stdin)) {

send\_len = strlen(send\_str);

/\* send string to multicast address \*/

if ((sendto(sock, send\_str, send\_len, 0,

(struct sockaddr \*) &mc\_addr,

sizeof(mc\_addr))) != send\_len) {

perror("sendto() sent incorrect number of bytes");

exit(1);

}

/\* clear send buffer \*/

memset(send\_str, 0, sizeof(send\_str));

}

close(sock);

exit(0);

}

### MulticastClient2.java

import java.io.\*;

import java.net.\*;

import java.util.Vector;

public class MulticastClient2 extends Thread{

public static final String MCAST\_ADDR = "230.0.0.1"; //dir clase D valida, grupo al que nos vamos a unir

public static final int MCAST\_PORT = 9013;//puerto multicast

public static final int DGRAM\_BUF\_LEN=512; //tamanio del buffer

public void run(){

InetAddress group =null;

try{

group = InetAddress.getByName(MCAST\_ADDR);//intenta resolver la direccion

}catch(UnknownHostException e){

e.printStackTrace();

System.exit(1);

}

Vector d = new Vector();

boolean salta=true;

try{

MulticastSocket socket = new MulticastSocket(MCAST\_PORT); //socket multicast

socket.joinGroup(group);//se une al grupo

int cd=0;

while(salta){

byte[] buf = new byte[DGRAM\_BUF\_LEN];//crea arreglo de bytes

//crea el datagram packet a recibir

DatagramPacket recv = new DatagramPacket(buf,buf.length);

socket.receive(recv);// ya se tiene el datagram packet

System.out.println("Host remoto: "+recv.getAddress());

System.out.println("Puerto: "+ recv.getPort());

byte [] data = recv.getData(); //aqui no se entienden los datos

// se guarda en un vector los datos recibidos

System.out.println("Datos recibidos: " + new String(data));

//d.addElement(new String(recv.getData()));

//if(++cd==5)

// salta=false;

} //se convierten los datos///

//System.out.println("Vector");

//for(Object a:d){

// System.out.println((String)a);

//}

}catch(IOException e){

e.printStackTrace();

System.exit(2);

}

}//run

public static void main(String[] args) {

try{

MulticastClient2 mc2 = new MulticastClient2();

mc2.start();

}catch(Exception e){e.printStackTrace();}

}//main

}//class

### MulticastServer2.java

import java.net.\*;

import java.io.\*;

public class MulticastServer2 extends Thread{

//dir clase D valida, grupo al que nos vamos a unir

public static final String MCAST\_ADDR = "230.0.0.1”;

public static final int MCAST\_PORT = 9013;

public static final int DGRAM\_BUF\_LEN = 512;

public void run(){

String msg = "Hola"; // se cambiará para poner la ip de la maquina con lo siguiente

InetAddress group = null;

try{

//msg=InetAddress.getLocalHost().getHostAddress();

group = InetAddress.getByName(MCAST\_ADDR); //resolver dir multicast

}catch(UnknownHostException e){

e.printStackTrace();

System.exit(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*inicia loop\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for(;;){

try{

MulticastSocket socket = new MulticastSocket(MCAST\_PORT);

socket.joinGroup(group); // se configura para escuchar el paquete

DatagramPacket packet = new DatagramPacket(msg.getBytes(),msg.length(),group,MCAST\_PORT);

System.out.println("Enviando: " + msg+" con un TTL= "+socket.getTimeToLive());

socket.send(packet);

socket.close();

}catch(IOException e){

e.printStackTrace();

System.exit(2);

}

try{

Thread.sleep(1000\*5);

}catch(InterruptedException ie){}

}//for;;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*termina Loop\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}//run

public static void main(String[] args) {

try{

MulticastServer2 mc2 = new MulticastServer2();

mc2.start();

}catch(Exception e){e.printStackTrace();}

}//main

}//class

## 2.1.2 Sockets de datagrama no bloqueante

### Servidor eco UDP no bloqueante

import java.net.\*;

import java.io.\*;

import java.nio.\*;

import java.nio.channels.\*;

import java.util.\*;

public class SEcoUDPNB{

public final static int PUERTO = 7;

public final static int TAM\_MAXIMO = 65507;

public static void main(String[] args) {

int port = PUERTO;

try{

DatagramChannel canal = DatagramChannel.open();

canal.configureBlocking(false);

DatagramSocket socket = canal.socket();

SocketAddress dir = new InetSocketAddress(port);

socket.bind(dir);

Selector selector = Selector.open();

canal.register(selector,SelectionKey.OP\_READ);

ByteBuffer buffer = ByteBuffer.allocateDirect(TAM\_MAXIMO);

while(true){

selector.select(5000);

Set sk = selector.selectedKeys();

Iterator it = sk.iterator();

while(it.hasNext()){

SelectionKey key = (SelectionKey)it.next();

it.remove();

if(key.isReadable()){

buffer.clear();

SocketAddress client = canal.receive(buffer);

buffer.flip();

int eco = buffer.getInt();

if(eco==100){

canal.close();

System.exit(0);

}else{

System.out.println("Dato leido: "+eco);

buffer.flip();

canal.send(buffer,client);

}//else

}//if

}//while2

}//while

}catch(IOException e){

System.err.println(e);

}//catch

}//main

}//class

### Cliente eco UDP no bloqueante

import java.net.\*;

import java.io.\*;

import java.nio.\*;

import java.nio.channels.\*;

import java.util.\*;

public class CEcoUDPNB{

public final static int PUERTO = 7;

private final static int LIMITE = 100;

public static void main(String[] args) {

boolean bandera=false;

SocketAddress remoto = new InetSocketAddress("127.0.0.1",PUERTO);

try{

DatagramChannel canal = DatagramChannel.open();

canal.configureBlocking(false);

canal.connect(remoto);

Selector selector = Selector.open();

canal.register(selector,SelectionKey.OP\_WRITE);

ByteBuffer buffer = ByteBuffer.allocateDirect(4);

int n = 0;

while(true){

selector.select(5000); //espera 5 segundos por la conexión

Set sk = selector.selectedKeys();

if(sk.isEmpty() && n == LIMITE || bandera){

canal.close();

break;

}else{

Iterator it = sk.iterator();

while(it.hasNext()){

SelectionKey key = (SelectionKey)it.next();

it.remove();

if(key.isWritable()){

buffer.clear();

buffer.putInt(n);

buffer.flip();

canal.write(buffer);

System.out.println("Escribiendo el dato: "+n);

n++;

if(n==LIMITE){

//todos los paquetes han sido escritos;

buffer.clear();

buffer.putInt(100);

buffer.flip();

canal.write(buffer);

bandera = true;

key.interestOps(SelectionKey.OP\_READ);

break;

}//if

}//if

}//while

}//else

}//while

}catch(Exception e){

System.err.println(e);

}//catch

}//main

}//class

## 4.1.a

### Hijo.c

#include <sys/types.h>

#include <stdlib.h>

#include <stdio.h>

main(){

int i = 0;

switch (fork()){

case -1:

perror("Error al crear procesos");

exit (-1);

break;

case 0: //Código para el hijo

while(i<10){

sleep(1);

printf("\t\tSoy el proceso hijo: %d\n", i++);

}

break;

default: //Código para el padre

while(i<10){

printf("Soy el proceso padre: %d\n", i++);

sleep(2);

}

};

exit (0);

}

## 4.1.b

### canceHijo1.c

#include <sys/types.h>

#include <sys/wait.h>

#include <stdlib.h>

#include <stdio.h>

#define NUM\_PROCESOS 5

int I = 0;

void codigo\_del\_proceso(int id){

int i;

for(i=0; i< 50; i++)

printf("Proceso %d: i = %d, I = %d\n", id, i, I++);

//El valor de id se almacena en los bits 8 al 15

// antes de devolver al proceso padre

exit(id);

}

main(){

int id [NUM\_PROCESOS] = {1, 2, 3, 4, 5};

int p, pid, salida;

for(p=0; p < NUM\_PROCESOS; p++){

pid = fork();

if(pid == -1){

perror("Error al crear un proceso: ");

exit(-1);

} else if(pid == 0) //Codigo del proceso hijo

codigo\_del\_proceso(id [p]);

}

//Esta parte solo la ejecuta el proceso padre

for(p=0; p< NUM\_PROCESOS; p++){

pid = wait(&salida);

printf("Proceso %d con id = %x terminado\n", pid, salida >> 8);

//El id del proceso devuelto con la llamada a

// exit se almacena en los bits 8 al 15

}

}

## 4.1.c

### canceHijo2.c

// Compilar gcc -o hilos hilos.c –lpthread

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <pthread.h>

#define NUM\_HILOS 5

int I = 0;

void \*codigo\_del\_hilo(void \*id){

int i;

for(i=0; i< 50; i++)

printf("Hilo %d: i=%d, I=%d\n",\*(int \*)id, i, I++);

pthread\_exit(id);

}

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

int h;

pthread\_t hilos[NUM\_HILOS];

int id[NUM\_HILOS] = {1,2,3,4,5};

int error;

int \*salida;

for(h = 0; h < NUM\_HILOS; h++){

error = pthread\_create(&hilos[h], NULL, codigo\_del\_hilo, &id[h]);

if(error){

fprintf(stderr,"Error %d: %s\n", error, strerror (error));

exit (-1);

}

}

for(h=0; h< NUM\_HILOS; h++){

error = pthread\_join(hilos[h],(void \*\*)&salida);

if(error)

fprintf(stderr,"Error %d: %s\n", error, strerror(error));

else

printf("Hilo %d terminado\n", \*salida);

}

}

## 4.2.3 Serie de Taylor

### errores.h

#ifndef \_\_ERRORES\_\_

#define \_\_ERRORES\_\_

#include <stdio.h>

#define error\_fatal(codigo, texto) do{\

fprintf(stderr, "%s:%d: Error: %s - %s\n",\

\_\_FILE\_\_, \_\_LINE\_\_, texto, strerror(codigo));\

abort();\

} while (0)

#endif

### taylor.c

// Compilación gcc -o exp exp.c -lpthread

#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include "errores.h"

//Presenta el resultado final del cálculo

void fin\_del\_calculo(void \*arg){

double resultado = \*(double \*)arg;

printf("Resultado final: %g\n",resultado);

}

//Cálculo de la serie de Taylor

void \*calculo(void \*arg){

int error, estado\_ant, tipo\_ant;

double x = \*(double \*)arg, resultado = 1, sumando = 1;

long i, j;

//Parte obligatoria del cálculo

//Deshabilitamos la posibilidad de cancelar el hilo

error=pthread\_setcancelstate(PTHREAD\_CANCEL\_DISABLE, &estado\_ant);

if (error) error\_fatal(error, "pthread\_setcancelstate");

for(i = 1; i<10; i++){

sumando \*= x/i;

resultado += sumando;

}

printf("Primera aproximación de exp(%g) = %g\n", x, resultado);

pthread\_cleanup\_push(fin\_del\_calculo, &resultado);

/\*Una vez ejecutada la parte obligatoria habilitamos

la posibilidad de cancelar el hilo \*/

error = pthread\_setcanceltype(PTHREAD\_CANCEL\_DEFERRED,&tipo\_ant);

if(error) error\_fatal(error, "pthread\_setcanceltype");

error = pthread\_setcancelstate(PTHREAD\_CANCEL\_ENABLE,&estado\_ant);

if(error) error\_fatal(error, "pthread\_setcancelstate");

//En esta parte de refinamiento del cálculo se aceptan peticiones de cancelación

printf("Refinamiento del cálculo\n");

for(;;){

pthread\_testcancel(); //Punto de cancelación

//Si no hay petición de cancelación se ejecuta un nuevo refinamiento del cálculo

for(j=0; j<10; j++,i++){

sumando \*= x/i;

resultado += sumando;

}

}

pthread\_cleanup\_pop(1);

}

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

pthread\_t hilo;

int error, plazo;

double x;

//Análisis de los argumentos

if(argc != 3){

printf("Forma de uso: %s x plazo\n", argv[0]);

exit(-1);

} else {

x = atof(argv[1]);

plazo = atoi(argv[2]);

}

//Creación del hilo que realiza el cálculo

error = pthread\_create(&hilo, NULL, calculo, &x);

if(error) error\_fatal(error, "pthread\_create");

//una vez concluido el plazo se cancela el hilo que calcula

sleep(plazo);

error = pthread\_cancel(hilo);

if(error) error\_fatal(error, "pthread\_cancel");

//Esperamos hasta que la cancelación se haga efectiva

error = pthread\_join(hilo, NULL);

if(error) error\_fatal(error, "pthread\_join");

}

## 4.2.4.a Cámbio de atributos

## atributos.c

// Compilación: gcc -o atributo atributos.c -lpthread

#include <pthread.h>

#include <limits.h>

#include "errores.h"

#include <stdio.h>

#include <stdlib.h>

void \*codigo\_del\_hilo(void \*arg){

pthread\_attr\_t \*atributos = arg;

int detachstate;

int tam\_pila;

int error;

error = pthread\_attr\_getdetachstate(atributos, &detachstate);

if(error)

error\_fatal(error,"pthread\_attr\_getdetachstate");

else if (detachstate == PTHREAD\_CREATE\_DETACHED)

printf("Hilo separado\n");

else

printf("Hilo NO separado\n");

error = pthread\_attr\_getstacksize(atributos, &tam\_pila);

if(error)

error\_fatal(error, "pthread\_attr\_getstacksize");

else

printf("Hilo. Tamaño de la pila : %d bytes = %d x %d\n",tam\_pila,

tam\_pila/PTHREAD\_STACK\_MIN, PTHREAD\_STACK\_MIN);

return NULL; //Equivalente a pthread\_exit(NULL)

}

int main(){

pthread\_t hilo;

pthread\_attr\_t atributos;

size\_t tam\_pila;

int error;

//Inicialización de los atributos

error = pthread\_attr\_init(&atributos);

if (error) error\_fatal(error, "pthread\_attr\_ini");

//Activación del atributo PTHREAD\_CREATE\_DETACHED

error = pthread\_attr\_setdetachstate(&atributos, PTHREAD\_CREATE\_DETACHED);

if(error) error\_fatal(error, "pthread\_attr\_sedetachstate");

//Manipulación del tamaño de la pila

error = pthread\_attr\_getstacksize(&atributos, &tam\_pila);

if(error) error\_fatal(error,"pthread\_attr\_getstacksize");

else{

printf("tamaño de la pila por defecto: %d bytes\n",tam\_pila);

printf("Tamaño mínimo de la pila: %d bytes\n",PTHREAD\_STACK\_MIN);

}

error = pthread\_attr\_setstacksize(&atributos, 3\*PTHREAD\_STACK\_MIN);

if(error) error\_fatal(error,"pthread\_create");

//Creamos el hilo con los atributos anteriores

error = pthread\_create(&hilo, &atributos, codigo\_del\_hilo, &atributos);

if(error) error\_fatal(error, "pthread\_create");

printf("Fin del hilo pricipal.\n");

pthread\_exit(NULL);

}

## 4.2.4.b

### Hilo\_retorno.c

#include <pthread.h>

#include <stdio.h>

#include <malloc.h>

struct sumandos{

int a;

int b;

};

void\* suma (void\* arg){

int res = 0;

struct sumandos \*datos = (struct sumandos \*)arg;

printf("\nDato a: %d",datos->a);

printf("\nDato b: %d",datos->b);

res = datos->a + datos->b;

return (void \*)res;

}

int main (){

pthread\_t thread;

int resultado;

struct sumandos \*x = (struct sumandos \*)

malloc(sizeof(struct sumandos));

x->a=3;

x->b=2;

pthread\_create(&thread, NULL, &suma,x);

pthread\_join(thread,(void \*)&resultado);

printf("\nEl resultado de sumar %d y %d es %d.\n",

x->a, x->b,resultado);

free(x);

return 0;

}

## 4.2.4.c Hilos y sockets

### ecoHilosServidor.c

#include <fcntl.h>

#include <string.h>

#include <stdlib.h>

#include <errno.h>

#include <stdio.h>

#include <netinet/in.h>

#include <resolv.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <pthread.h>

void\* SocketHandler(void\*);

int main(int argv, char\*\* argc){

int host\_port= 1101;

struct sockaddr\_in my\_addr;

int hsock;

int \* p\_int ;

int err;

socklen\_t addr\_size = 0;

int\* csock;

struct sockaddr\_in sadr;

pthread\_t thread\_id=0;

hsock = socket(AF\_INET, SOCK\_STREAM, 0);

if(hsock == -1){

printf("Error initializando el socket %d\n", errno);

goto FINISH;

}

p\_int = (int\*)malloc(sizeof(int));

\*p\_int = 1;

if( (setsockopt(hsock, SOL\_SOCKET, SO\_REUSEADDR, (char\*)p\_int, sizeof(int)) == -1 )||

(setsockopt(hsock, SOL\_SOCKET, SO\_KEEPALIVE, (char\*)p\_int, sizeof(int)) == -1 ) ){

printf("Error configurando opciones %d\n", errno);

free(p\_int);

goto FINISH;

}

free(p\_int);

my\_addr.sin\_family = AF\_INET ;

my\_addr.sin\_port = htons(host\_port);

memset(&(my\_addr.sin\_zero), 0, 8);

my\_addr.sin\_addr.s\_addr = INADDR\_ANY ;

if( bind( hsock, (struct sockaddr\*)&my\_addr, sizeof(my\_addr)) == -1 ){

fprintf(stderr,"Error ligando el socket %d\n",errno);

goto FINISH;

}

if(listen( hsock, 10) == -1 ){

fprintf(stderr, "Error en el listen%d\n",errno);

goto FINISH;

}

addr\_size = sizeof(struct sockaddr\_in);

for(;;){

printf("Esperando una conexión\n");

csock = (int\*)malloc(sizeof(int));

if((\*csock = accept( hsock, (struct sockaddr\*)&sadr, &addr\_size))!= -1){

printf("\Conexión recibida desde: %s\n",inet\_ntoa(sadr.sin\_addr));

pthread\_create(&thread\_id,0,&SocketHandler, (void\*)csock );

pthread\_detach(thread\_id);

}

else{

fprintf(stderr, "Error en el accept %d\n", errno);

}

}

FINISH:

;

}

void\* SocketHandler(void\* lp){

int \*csock = (int\*)lp;

char buffer[1024];

int buffer\_len = 1024;

int bytecount;

memset(buffer, 0, buffer\_len);

if((bytecount = recv(\*csock, buffer, buffer\_len, 0))== -1){

fprintf(stderr, "Error recibiendo datos %d\n", errno);

goto FINISH;

}

printf("Bytes recibidos %d\nCadena recibida: \"%s\"\n", bytecount, buffer);

strcat(buffer, " ECO DEL SERVIDOR");

if((bytecount = send(\*csock, buffer, strlen(buffer), 0))== -1){

fprintf(stderr, "Error recibiendo datos %d\n", errno);

goto FINISH;

}

printf("Bytes enviados: %d\n", bytecount);

FINISH:

free(csock);

return 0;

}

### ecoHilosCliente.c

#include <fcntl.h>

#include <string.h>

#include <stdlib.h>

#include <errno.h>

#include <stdio.h>

#include <netinet/in.h>

#include <resolv.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

int main(int argv, char\*\* argc){

int host\_port= 1101;

char\* host\_name="127.0.0.1";

struct sockaddr\_in my\_addr;

char buffer[1024];

int bytecount;

int buffer\_len=0;

int hsock;

int \* p\_int;

int err;

hsock = socket(AF\_INET, SOCK\_STREAM, 0);

if(hsock == -1){

printf("Error initializing socket %d\n",errno);

goto FINISH;

}

p\_int = (int\*)malloc(sizeof(int));

\*p\_int = 1;

if( (setsockopt(hsock, SOL\_SOCKET, SO\_REUSEADDR, (char\*)p\_int, sizeof(int)) == -1 )||

(setsockopt(hsock, SOL\_SOCKET, SO\_KEEPALIVE, (char\*)p\_int, sizeof(int)) == -1 ) ){

printf("Error setting options %d\n",errno);

free(p\_int);

goto FINISH;

}

free(p\_int);

my\_addr.sin\_family = AF\_INET ;

my\_addr.sin\_port = htons(host\_port);

memset(&(my\_addr.sin\_zero), 0, 8);

my\_addr.sin\_addr.s\_addr = inet\_addr(host\_name);

if( connect( hsock, (struct sockaddr\*)&my\_addr, sizeof(my\_addr)) == -1 ){

if((err = errno) != EINPROGRESS){

fprintf(stderr, "Error connecting socket %d\n", errno);

goto FINISH;

}

}

buffer\_len = 1024;

memset(buffer, '\0', buffer\_len);

printf("Enter some text to send to the server (press enter)\n");

fgets(buffer, 1024, stdin);

buffer[strlen(buffer)-1]='\0';

if( (bytecount=send(hsock, buffer, strlen(buffer),0))== -1){

fprintf(stderr, "Error sending data %d\n", errno);

goto FINISH;

}

printf("Sent bytes %d\n", bytecount);

if((bytecount = recv(hsock, buffer, buffer\_len, 0))== -1){

fprintf(stderr, "Error receiving data %d\n", errno);

goto FINISH;

}

printf("Recieved bytes %d\nReceived string \"%s\"\n", bytecount, buffer);

close(hsock);

FINISH:

;

}

## 4.3.3.a Candados

### Candados.c

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

void \*f();

pthread\_mutex\_t m1 = PTHREAD\_MUTEX\_INITIALIZER;

int cont = 0;

main(){

int r1, r2;

pthread\_t t1, t2;

if(r1 = pthread\_create(&t1, NULL, f, NULL)){

printf("Error al crear el hilo t1\n");

}

if(r2 = pthread\_create(&t2, NULL, f, NULL)){

printf("Error al crear el hilo t2\n");

}

pthread\_join(t1,NULL);

pthread\_join(t2,NULL);

exit(0);

}

void \*f(){

pthread\_mutex\_lock(&m1);

cont++;

printf("El valor del contador es: %d\n",cont);

pthread\_mutex\_unlock(&m1);

}

## 4.3.3.b variables de condición

### Cond.c

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

pthread\_mutex\_t m =

PTHREAD\_MUTEX\_INITIALIZER;

pthread\_cond\_t cv = PTHREAD\_COND\_INITIALIZER;

void \* f1();

void \* f2();

int cont = 0;

#define LIM 10

#define L1 3

#define L2 6

main(){

pthread\_t t1, t2;

pthread\_create(&t1, NULL, f1, NULL);

pthread\_create(&t2, NULL, f2, NULL);

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

printf("La cuenta final es: %d\n", cont);

exit(0);

}

void \*f1(){

for(;;){

pthread\_mutex\_lock(&m);

pthread\_cond\_wait(&cv, &m);

cont++;

printf("cont inc. desde f1(): %d\n",cont);

pthread\_mutex\_unlock(&m);

if(cont >= LIM) return;

}

}

void \*f2(){

for(;;){

pthread\_mutex\_lock(&m);

if(cont< L1 || cont > L2)

pthread\_cond\_signal(&cv);

else{

cont++;

printf("cont inc. desde f2(): %d\n",cont);

}

pthread\_mutex\_unlock(&m);

if(cont >= LIM) return;

}

}

## 4.3.3.c Semáforos

### Semaforo.c

#include <semaphore.h>

#include <pthread.h>

#include <stdio.h>

#define HILOS 20

sem\_t okComprarLeche;

int lecheDisponible;

void\* comprador(void \*arg){

sem\_wait(&okComprarLeche);

if(!lecheDisponibles){

// Comprar algo de leche

++lecheDisponible;

}

sem\_post(&okComprarLeche);

}

int main(int argc, char \*\*argv)

{

pthread\_t hiloss[HILOS];

int i;

lecheDisponible = 0;

// Inicializamos el semáforo

if(sem\_init(&okComprarLeche, 0, 1)){

printf("No se pudo inicializar el semáforo\n");

return -1;

}

for(i = 0; i < HILOS; ++i){

if(pthread\_create(&hilos[i], NULL, &comprador, NULL)){

printf("No se pudo crear el hilo número %d\n", i);

return -1;

}

}

for(i = 0; i < HILOS; ++i){

if(pthread\_join(hilos[i], NULL)){

printf("No se pudo ligar el hilo número %d\n", i);

return -1;

}

}

sem\_destroy(&okComprarLeche);

// Asegurémonos de no tener demasiada leche.

printf("Total de leche: %d\n", lecheDisponible);

return 0;

}

## 4.3.3.d Tuberias

### Tuberías.c

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <pthread.h>

int fd[2];

void \*lector(){

while(1){

char ch;

int resultado;

resultado = read (fd[0],&ch,1);

if (resultado != 1) {

perror("Error en read");

exit(3);

} printf ("Lector: %c\n", ch); }

}

void \*escritor\_ABC(){

int resultado;

char ch='A';

while(1){

resultado = write (fd[1], &ch,1);

if (resultado != 1){

perror ("Error en write");

exit (2);

}

printf ("Escritor\_ABC: %c\n", ch);

if(ch == 'Z')

ch = 'A'-1;

ch++;

}

}

void \*escritor\_abc(){

int resultado;

char ch='a';

while(1){

resultado = write (fd[1], &ch,1);

if (resultado != 1){

perror ("Error en write");

exit (2);

}

printf ("Escritor\_abc: %c\n", ch);

if(ch == 'z')

ch = 'a'-1;

ch++;

}

}

int main(){

pthread\_t tid1,tid2,tid3;

int resultado;

resultado = pipe (fd);

if (resultado < 0){

perror(“Error en pipe ");

exit(1);

}

pthread\_create(&tid1,NULL,lector,NULL);

pthread\_create(&tid2,NULL,escritor\_ABC,NULL);

pthread\_create(&tid3,NULL,escritor\_abc,NULL);

pthread\_join(tid1,NULL);

pthread\_join(tid2,NULL);

pthread\_join(tid3,NULL);

}

## 4.3.3.e EcoHilos java

### SEcoHilos.java

import java.io.\*;

import java.net.\*;

public class SEcoHilos {

public static void main(String[] args){

try{

ServerSocket s = new ServerSocket(9000);

System.out.println("Servidor listo en el puerto"+ s.getLocalPort());

for(;;){

Socket cl = s.accept();

System.out.println("Cliente conectado..\n");

Manejador m = new Manejador(cl);

m.start();

}//for

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

class Manejador extends Thread{

Socket cl;

public Manejador(Socket cl){

this.cl = cl;

}//constructor

public void run(){

try{

PrintWriter pw = new PrintWriter(new OutputStreamWriter(cl.getOutputStream()));

BufferedReader br = new BufferedReader(new InputStreamReader(cl.getInputStream()));

String linea="";

for(;;){

linea= br.readLine();

System.out.println("Recibiendo mensaje: "+linea);

if(linea.indexOf("SALIR")>=0){

System.out.println("Cliente se va...");

cl.close();

break;

}

pw.println(linea);

pw.flush();

}//for

}catch(IOException io){

io.printStackTrace();

}//catch

}//run

}

### CEcoHilos.java

import java.net.\*;

import java.io.\*;

public class CEcoHilos {

public static void main(String[] args){

try{

InetAddress srv = InetAddress.getByName("127.0.0.1");

Socket cl = new Socket(srv,9000);

PrintWriter pw = new PrintWriter(new OutputStreamWriter(cl.getOutputStream()));

BufferedReader br = new BufferedReader(new InputStreamReader(cl.getInputStream()));

BufferedReader br2 = new BufferedReader(new InputStreamReader(System.in));

String linea="";

System.out.println("Escribe mensajes <ENTER> para enviar, <SALIR> para terminar\n");

for(;;){

linea= br2.readLine();

pw.println(linea);

pw.flush();

if(linea.indexOf("SALIR")>=0){

System.out.println("Adios...");

cl.close();

System.exit(0);

}//if

String eco=br.readLine();

System.out.println("ECO: "+eco);

}//for

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

## 4.3.3.f Mutex java

### Mutex.java

import java.util.concurrent.locks.ReentrantLock;

public class Mutex implements Runnable {

int cont;

ReentrantLock rl;

public Mutex(){

this.cont=0;

rl= new ReentrantLock();

}//Mutex

int getCont(){

return this.cont;

}

public void run(){

System.out.println("Comienza mutex....");

rl.lock(); /////

int tmp= cont;

try{

Thread.sleep(100);

}catch(InterruptedException ie){}

try{

tmp ++;

cont=tmp;

}catch(Exception e){

}finally{

rl.unlock(); /////

}

}//run

public static void main(String[] args){

try{

Mutex m = new Mutex();

Thread t1 = new Thread(m);

Thread t2 = new Thread(m);

Thread t3 = new Thread(m);

Thread t4 = new Thread(m);

Thread t5 = new Thread(m);

t1.start(); t2.start(); t3.start(); t4.start(); t5.start();

t1.join(); t2.join(); t3.join(); t4.join(); t5.join();

System.out.println("Cont: "+ m.getCont());

}catch(Exception e){

e.printStackTrace();

}//catch

}//main

}

## 4.3.3.g Variables de condición en java

### CondDemos.java

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

public class CondDemo{

public static void main(String[] args) {

Shared s = new Shared();

new Producer(s).start();

new Consumer(s).start();

}

}

class Shared {

private volatile char c;

private volatile boolean available;

private final Lock lock;

private final Condition condition;

Shared() {

c = '\u0000'; //valor nulo

available = false;

lock = new ReentrantLock();

condition = lock.newCondition();

}

Lock getLock(){

return lock;

}

char getSharedChar() {

lock.lock();

try

{

while (!available)

try {

condition.await();

} catch (InterruptedException ie) {

ie.printStackTrace();

}

available = false;

condition.signal();

} finally {

lock.unlock();

return c;

}

}

void setSharedChar(char c) {

lock.lock();

try {

while (available)

try {

condition.await();

} catch (InterruptedException ie) {

ie.printStackTrace();

}

this.c = c;

available = true;

condition.signal();

} finally {

lock.unlock();

}

}

}

### Producer.java

import java.util.concurrent.locks.Lock;

class Producer extends Thread {

private final Lock l;

private final Shared s;

Producer(Shared s) {

this.s = s;

l = s.getLock();

}

@Override

public void run() {

for (char ch = 'A'; ch <= 'Z'; ch++) {

l.lock();

s.setSharedChar(ch);

System.out.println(ch + " Productor.");

l.unlock();

}

}

}

### Consumer.java

import java.util.concurrent.locks.Lock;

class Consumer extends Thread {

private final Lock l;

private final Shared s;

Consumer(Shared s) {

this.s = s;

l = s.getLock();

}

@Override

public void run() {

char ch;

do {

l.lock();

ch = s.getSharedChar();

System.out.println(ch + " Consumidor.");

l.unlock();

}

while (ch != 'Z');

}

}

## 4.3.3.h Semáforos

### Restaurante.java

Import import java.util.concurrent.Semaphore;

public class Restaurante {

private Semaphore mesas;

public Restaurante(int contadorMesas) {

// Crea un semaforo con las mesas que tenemos

this.mesas = new Semaphore(contadorMesas);

}

public void obtenerMesa(int idCliente) {

try {

System.out.println("Cliente #" + idCliente + " esta intentando obtener una mesa.");

// Adquiere un permiso para una tabla

mesas.acquire();

System.out.println("Cliente #" + idCliente + " consiguio una mesa.");

}

catch (InterruptedException ie) {

ie.printStackTrace();

}

}

public void regresaMesa(int idCliente) {

System.out.println("Cliente #" + idCliente + " devolvio mesa.");

mesas.release();

}

public static void main(String[] args) {

Restaurante r = new Restaurante(2);

for (int i = 1; i <= 5; i++) {

Cliente c = new Cliente(r, i);

c.start();

}

}

}

### Cliente.java

import java.util.Random;

class Cliente extends Thread {

private Restaurante r;

private int idCliente;

private static final Random aleatorio = new Random();

public Cliente(Restaurante r, int idCliente) {

this.r = r;

this.idCliente = idCliente;

}

public void run() {

r.obtenerMesa(this.idCliente);

try {

// Come durante un tiempo. Usa valores entre 1 y 30 segundos

int tiempoComida = aleatorio.nextInt(30) + 1 ;

System.out.println("Cliente #" + this.idCliente +" comera por " +

| tiempoComida + " segundos.");

Thread.sleep(tiempoComida \* 1000);

System.out.println("Cliente #" + this.idCliente + " termino de comer.");

} catch(InterruptedException ie) {

ie.printStackTrace();

}finally {

r.regresaMesa(this.idCliente);

}

}

}

## 4.3.3.i Tuberías

### Consumidor.java

import java.io.\*;

class Consumidor extends Thread{

private double promedio\_anterior=0;

private DataInputStream entrada;

public Consumidor(InputStream is){

entrada=new DataInputStream(is);

}//constructor

public void run(){

for(;;){

try{

double prom = entrada.readDouble();

if(Math.abs(prom-promedio\_anterior)>0.01){

System.out.println("El promedio actual es "+prom);

promedio\_anterior=prom;

}//if

}catch(IOException io){

io.printStackTrace();

}//catch

}//for

}//run

}//Consumidor

public class Tuberias {

public static void main(String[] args){

try{

PipedOutputStream po1 = new PipedOutputStream();

PipedInputStream pi1 = new PipedInputStream(po1);

PipedOutputStream po2 = new PipedOutputStream();

PipedInputStream pi2 = new PipedInputStream(po2);

Productor p= new Productor(po1);

Filtro f = new Filtro(pi1,po2);

Consumidor c = new Consumidor(pi2);

p.start(); f.start(); c.start();

} catch(IOException io){

io.printStackTrace();

}

}//main

}//class

### Productor.java

import java.io.\*;

import java.util.Random;

class Productor extends Thread{

private DataOutputStream salida;

private Random aleatorio = new Random();

public Productor(OutputStream os){

salida = new DataOutputStream(os);

}//constructor

@Override

public void run(){

while(true){

try{

double num = aleatorio.nextDouble();

salida.writeDouble(num);

salida.flush();

sleep(Math.abs(aleatorio.nextInt()%1000));

}catch(Exception e){

e.printStackTrace();

}//catch

}//while

}//run

}//productor

class Filtro extends Thread{

private DataInputStream entrada;

private DataOutputStream salida;

private double total=0 ;

private int cuenta=0;

public Filtro(InputStream is,OutputStream os){

entrada = new DataInputStream(is);

salida = new DataOutputStream(os);

}//constructor

public void run(){

for(;;){

try{

double x = entrada.readDouble();

total +=x;

cuenta++;

if(cuenta!=0){

salida.writeDouble(total/cuenta);

salida.flush();

}

}catch(IOException io){

io.printStackTrace();

}//catch

}//for

}//run

}//Filtro

## 4.5 Alberca de hilos

### AlbercaHilos.java

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class AlbercaHilos{

public static void main (String args[]){

System.out.println("Comienza la ejecución");

ExecutorService ex = Executors.newFixedThreadPool(10);

TareaAlbercaHilos t;

for(int i = 0;i<200; i++){

t = new TareaAlbercaHilos(""+i);

ex.execute(t);

}

ex.shutdown();

}

}

### TareaAlbercaHilos.java

public class TareaAlbercaHilos implements Runnable{

private int sleepTime;

private String name;

public TareaAlbercaHilos(String name){

this.name = name;//le asignamos un nombre a cada tarea.

sleepTime = 1000;

}

public void run(){

try{

System.out.printf("El hilo de la tarea "+this.name+" va a dormir durante %d milisegundos.\n",sleepTime);

Thread.sleep(sleepTime);//hacemos que cada hilo duerma durante 1 segundo

}catch(InterruptedException exception){

exception.printStackTrace();

}

System.out.println("Este hilo ya ha dormido bastante");

}

}

## 5.2 RMI

### Suma.java

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Suma extends Remote {

int suma(int a,int b) throws RemoteException;

}

### Cliente.java

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Cliente {

private Cliente() {}

public static void main(String[] args) {

String host = (args.length < 1) ? null : args[0];

try {

Registry registry = LocateRegistry.getRegistry(host);

//también puedes usar getRegistry(String host, int port)

Suma stub = (Suma) registry.lookup("Suma");

int x=5,y=4;

int response = stub.suma(x,y);

System.out.println("respuesta sumar "+x+" y "+y+" : " + response);

} catch (Exception e) {

System.err.println("Excepción del cliente: " +e.toString());

e.printStackTrace();

}

}

}

### Servidor.java

import java.rmi.registry.Registry;

import java.rmi.registry.LocateRegistry;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

//import Suma.\*;

public class Servidor implements Suma {

public Servidor() {}

public int suma(int a, int b) {

return a+b;

}

public static void main(String args[]) {

try {

//puerto default del rmiregistry

java.rmi.registry.LocateRegistry.createRegistry(1099);

System.out.println("RMI registro listo.");

} catch (Exception e) {

System.out.println("Excepcion RMI del registry:");

e.printStackTrace();

}//catch

try {

System.setProperty("java.rmi.server.codebase","file:/c:/Temp/Suma/");

Servidor obj = new Servidor();

Suma stub = (Suma) UnicastRemoteObject.exportObject(obj, 0);

// Ligamos el objeto remoto en el registro

Registry registry = LocateRegistry.getRegistry();

registry.bind("Suma", stub);

System.err.println("Servidor listo...");

} catch (Exception e) {

System.err.println("Excepción del servidor: " + e.toString());

e.printStackTrace();

}

}

}