Instituto Politécnico Nacional.

Escuela Superior de Cómputo.

Practica 1

Alumnos: ALLAN ULISES ZEPEDA IBARRA

JONATHAN ARCOS AYALA

3CM4

**Objetivo**

El estudiante implementará una aplicación para el envío de múltiples archivos a través de la red haciendo uso de sockets de flujo.

**Introducción**

Comenzaremos por instalar Java, para esto necesitamos descargar el JDK de la pagina oficial de java, nosotros descargaremos el JDK 7.79, para instalarlo se ejecutaran los siguientes comandos linux.

sudo apt-get purge openjdk-\\*

sudo mkdir -p /usr/local/java

Se copian lkos binarios

cd /home/**“tu\_nombre\_de\_usuario”**/Downloads

sudo cp -r jdk-8u20-linux-i586.tar.gz /usr/local/java/

cd /usr/local/java

sudo tar xvzf jdk-8u20-linux-i586.tar.gz

Editamos el path

sudo gedit /etc/profile

JAVA\_HOME=/usr/local/java/**jdk1.8.0\_20**  
 **PATH=$PATH:$HOME/bin:$JAVA\_HOME/bin**  
 **export JAVA\_HOME**  
 **export PATH**

sudo update-alternatives --install "/usr/bin/java" "java" "/usr/local/java/jdk1.8.0\_20/bin/java" 1

sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/local/java/jdk1.8.0\_20/bin/javac" 1

sudo update-alternatives --install "/usr/bin/javaws" "javaws" "/usr/local/java/jdk1.8.0\_20/bin/javaws" 1

sudo update-alternatives --set java /usr/local/java/jdk1.8.0\_20/bin/java

sudo update-alternatives --set javac /usr/local/java/jdk1.8.0\_20/bin/javac

sudo update-alternatives --set javaws /usr/local/java/jdk1.8.0\_20/bin/javaws

Y reinicias el sistema operativo.

Ya instalado eso tenemos que instalar el netbeans, que se descarga de la pagina oficial de netbeans. Y ejecutamos los siguientes códigos linux.

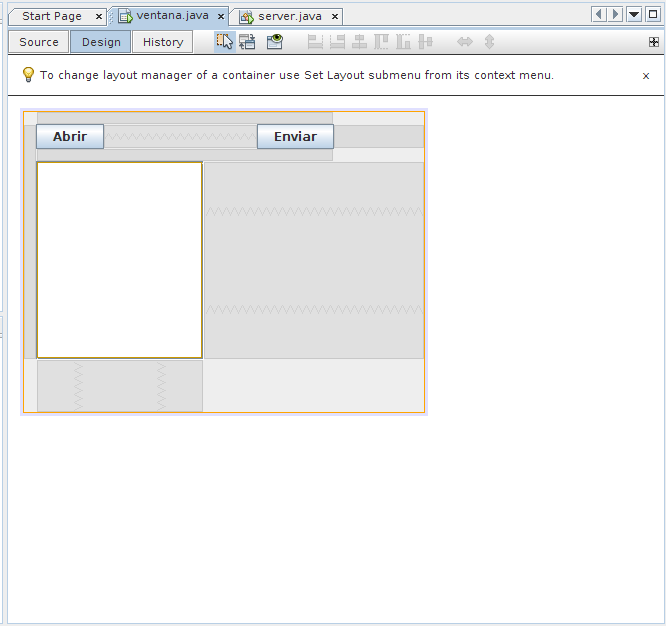
cd /home/usuario/Descargas

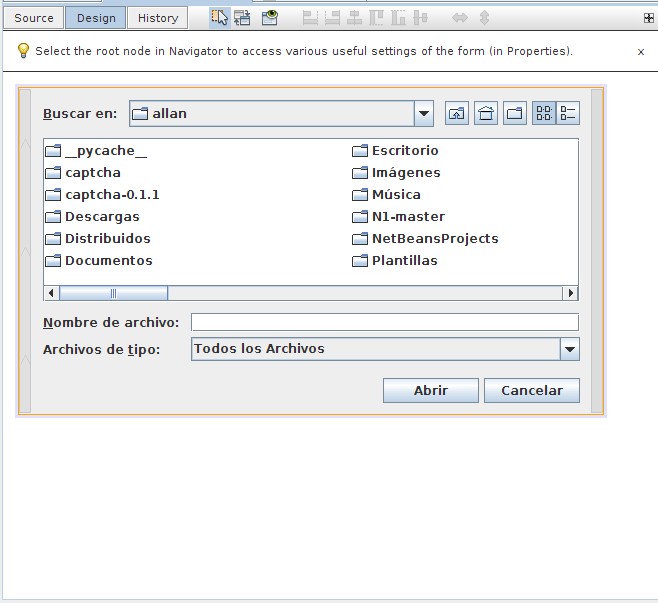
chmod 755 archivoInstalador.sh

sh archivoInstalador.sh

**Desarrollo**

Primeramente para hacer esta practica tenemos que hacer la interface grafica de usuario que consiste en 2 ventanas, la ventana principal donde se pueden ver en una lista los archivos con su peso así como el botón para abrir mas archivos y el botón de envío al servidor.

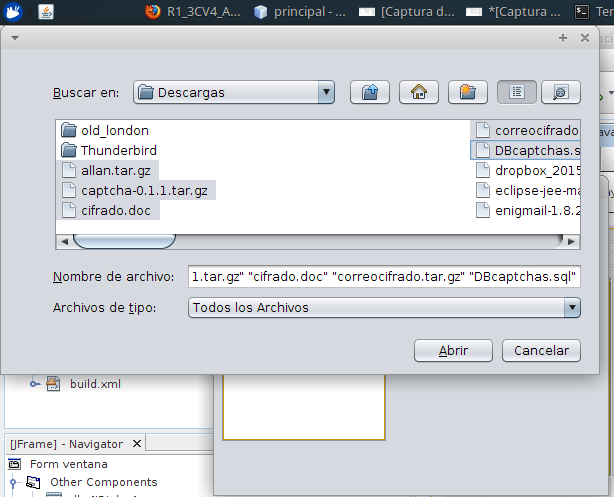




Para seleccionar un archivo creamos un evento en la ventana y dentro de este los imprimiremos en la lista

private void D1ActionPerformed(java.awt.event.ActionEvent evt) {   
 //pantalla de abrir archivo  
 JFileChooser selectorArchivo;  
 selectorArchivo = (JFileChooser)evt.getSource();  
 String coman;  
 coman = evt.getActionCommand();  
   
 if (coman.equals(JFileChooser.APPROVE\_SELECTION))  
 {  
 //se traen los archivos seleccionados a un arreglo de archivos  
 File[] arcSele = selectorArchivo.getSelectedFiles();  
 for(int i=0;i<(arcSele.length);i++)  
 {  
 //se agregan a vectores, el nombre, la direccion, el tamaño y se ponen para imprimirse en una lista  
 elem.add(arcSele[i].getName());  
 dir.add(arcSele[i].getAbsolutePath());  
 tam.add(arcSele[i].length());  
 aux.add(arcSele[i].getName()+" \t "+arcSele[i].length()+"bytes");  
 }  
 //se limpia la lista y se imprime el vector en la lista  
 Li1.removeAll();  
 Li1.setListData(aux);  
 dlg.setVisible(false);  
 }else if (coman.equals(JFileChooser.CANCEL\_SELECTION))  
 {  
 dlg.setVisible(false);  
 }  
 }

Quedando así





El boton enviar abre la conexión al socket servidor para enviar los archivos a este tenemos que montar para esto un socket servidor, que crearemos con el siguiente código.

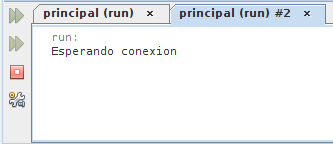
ServerSocket server;  
 Socket connection;  
   
 BufferedInputStream bis;  
 BufferedOutputStream bos = null;  
   
 byte[] receivedData;  
 int in, aux=-1, i=0;  
 long tam;  
 String file,n;  
   
 try{  
 //Servidor Socket en el puerto 5000  
 server = new ServerSocket( 5000 );  
 while ( true ) {  
 //Aceptar conexiones  
   
 if(i==0)  
 {  
 //recibe el numero de archivos  
 System.out.println("Esperando conexion");  
 //espera hasta que tengamos una conexion  
 connection = server.accept();  
 DataInputStream dis=new DataInputStream(connection.getInputStream());  
 //lee el numero de archivos  
 aux=dis.readInt();  
 System.out.println("Se enviaran "+aux+" archivos");  
 dis.close();  
 //incrementa en una unidad la bandera  
 i++;  
 }  
 else if(i==(aux))  
 {  
 //si la bandara es igual al numero de archivos recibidos prepara todo para terminar la recepcion  
 System.out.println(i);  
 //espera hasta que tenga una conexion  
 connection = server.accept();  
 //Buffer de 1024 bytes  
 //establece el tamaño de bloque  
 receivedData = new byte[1024];  
 bis = new BufferedInputStream(connection.getInputStream());  
 DataInputStream dis=new DataInputStream(connection.getInputStream());  
 //Recibimos el nombre del fichero  
 file = dis.readUTF();  
 System.out.println("Nombre del archivo: " + file);  
 //recibimos el tamaño del fichero  
 tam = dis.readLong();  
 System.out.println("Tamaño del archivo: " + tam + "mb");  
 file = file.substring(file.indexOf('\\')+1,file.length());  
 //Para guardar fichero recibido  
 bos = new BufferedOutputStream(new FileOutputStream(file));  
 while ((in = bis.read(receivedData)) != -1){  
 bos.write(receivedData,0,in);  
 }  
 bos.close();  
 dis.close();  
 //termina la transferencia y la prepara para recibir un nuevo paquete de archivos  
 i=0;  
 System.out.println("Transferencia terminada");  
 }  
 else  
 {  
 //recibe archivos  
 System.out.println(i);  
 connection = server.accept();  
 //Buffer de 1024 bytes  
 receivedData = new byte[1024];  
 bis = new BufferedInputStream(connection.getInputStream());  
 DataInputStream dis=new DataInputStream(connection.getInputStream());  
 //Recibimos el nombre del fichero  
 file = dis.readUTF();  
 System.out.println("Nombre del archivo: " + file);  
 tam = dis.readLong();  
 System.out.println("Tamaño del archivo: " + tam + "mb");  
 file = file.substring(file.indexOf('\\')+1,file.length());  
 //Para guardar fichero recibido  
 bos = new BufferedOutputStream(new FileOutputStream(file));  
 while ((in = bis.read(receivedData)) != -1){  
 bos.write(receivedData,0,in);  
 }  
 bos.close();  
 dis.close();  
 i++;  
 }  
 }  
 }catch(Exception e){  
 System.err.println(e);  
 }

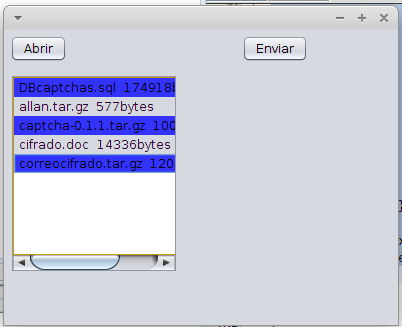
En este socket tenemos la recepción de un mensaje con el numero de archivos que se enviaran, y después uno a uno los archivos los guarda.

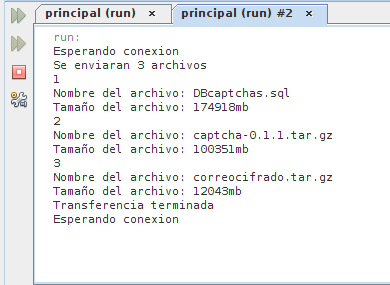
El envío lo ponemos en el botón de la interface el código que envía es el siguiente:

int [] seleccionados = Li1.getSelectedIndices();  
 //System.out.println(Arrays.toString(seleccionados));  
 DataInputStream input;  
 BufferedInputStream bis = null;  
 BufferedOutputStream bos;  
 int in;  
 byte[] byteArray;  
 String filename=null;  
 if(seleccionados.length>0){  
 try{  
 //envio de el numero de archivos  
 //se abre la conexion del socket  
 Socket client = new Socket("localhost", 5000);  
 bos = new BufferedOutputStream(client.getOutputStream());  
 //se prepara para mandar el numero de archivos  
 DataOutputStream dos=new DataOutputStream(client.getOutputStream());  
 dos.writeInt(seleccionados.length);  
 //delay para que termine de transferir  
 client.setSoLinger(true, 1000);  
 //se cierra la conexion  
 bos.close();  
 }catch(Exception e){  
 System.err.println(e);  
 System.out.println("Error de conexions");  
 JOptionPane.showMessageDialog(this, "No se pudo conectar al servidor");  
 }  
 try{  
 //envio de archivos  
 for(int i=0;i<seleccionados.length;i++) {  
 //se abre la conexion  
 Socket client = new Socket("localhost", 5000);  
 bos = new BufferedOutputStream(client.getOutputStream());  
 //se prepara para enviar el nombre y el tamaño del archivo  
 DataOutputStream dos=new DataOutputStream(client.getOutputStream());  
 dos.writeUTF(elem.elementAt(seleccionados[i]));  
 dos.writeLong(tam.elementAt(seleccionados[i]));  
 //se abren bariables para envia los archivos  
 filename=dir.elementAt(seleccionados[i]);  
 File localFile = new File( filename );  
 bis = new BufferedInputStream(new FileInputStream(localFile));  
 //tamaño de bloques a enviar  
 byteArray = new byte[8192];  
 //se escriben los archivos  
 while ((in = bis.read(byteArray)) != -1){  
 bos.write(byteArray,0,in);  
 }  
 // se cierra la conexion  
 client.setSoLinger(true, 1000);  
 bis.close();  
 bos.close();  
 }  
 }catch(Exception e){  
 System.err.println(e);  
 System.out.println("Error de conexions");  
 JOptionPane.showMessageDialog(this, "No se pudo conectar al servidor");  
 }  
 }else{  
 JOptionPane.showMessageDialog(this, "No seleccionaste ningun archivo de la lista");  
 }

El envío queda como se muestra a continuación.







**Cuestionario**

1. ¿Qué tipo de archivos se enviaron más rápido?

Los de menor peso como archivos de texto y de datos

2. ¿Cuál fue el número máximo de archivos que fue posible enviar a la vez?

Mas de 138

3. ¿Cuál fue el tamaño de archivo más grande que se pudo transferir? ¿por qué?

Se transfirieron archivos de 2GB sin problema

4. ¿Qué es el orden de red?

5. ¿Por qué razón es importante utilizar el orden de red al enviar los datos a través de un socket?

6. Si deseáramos enviar archivos de tamaño muy grande, ¿qué cambios sería necesario hacer con respecto a los tipos de datos usados para medir el tamaño de los archivos, así como para leer bloques de datos del archivo?

Cambiando el tamaño de bloque a enviar

**Conclusiones**

Arcos Ayala Jonathan: En esta practica podemos ver como utilizar los protocolos de flujo y su configuración para poder montar cualquier tipo servicio en las capas superiores.

Zepeda Ibarra Allan Ulises: Pues si bien es muy fácil la implementación de sockets en java, aun así no es tan simple lograr enviar exactamente lo que uno quiere.

**Bibliografía**

<http://www.programacion.com.py/escritorio/java-escritorio/sockets-en-java-udp-y-tcp>

<http://es.wikihow.com/instalar-Oracle-Java-JDK-en-Ubuntu-Linux>

<http://lignux.com/instalar-el-ultimo-netbeans-y-jdk/>