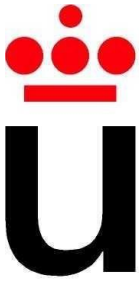
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA



Universidad
Rey Juan Carlos

ECHO CLIENT-SERVER

LENGUAJES DE PROGRAMACIÓN

Tarik Saghouani Ben-Khalek

Pablo Sánchez Gómez

Eva María Hoyo de la Cruz

José Ignacio Manso LLanes

Contenido

Introducción

Para realizar una conexión cliente-servidor entre dos máquinas, es preciso que estén comunicadas, para realizar esta conexión han sido usados sockets. Por medio de esta conexión en la que se define la ip, en nuestro caso se toma la local del equipo (localhost) ya que no se van a hacer pruebas con distintos equipos, además se tiene que configurar el puerto por el que se va a comunicar, esto ha de hacerse en ambas máquinas, en este caso en ambos ficheros, una vez hecho esto se envían y reciben mensajes entre ambos y se muestran por pantalla. La comunicación se mantiene establecida mientras el servidor está recibiendo mensajes por parte del cliente, una vez el cliente ya no manda más mensajes, se cierra su conexión y el servidor se mantiene a la espera de recibir nuevas conexiones por parte de otros clientes.

Entrega

Tal y como indica el enunciado de la practica esta entrega consta de dos partes. En la primera, en un archivo adjunto a este documento se encuentra el código del servidor, que además consta en el ANEXO I más adelante.
Adicionalmente, y para una mejor comprensión del código desarrollado por este grupo, una explicación de este.

Código

Siguiendo las instrucciones del enunciado, esta práctica consta de dos ficheros de código. El primero de ellos es el código correspondiente con el servidor y el segundo con el cliente.

Es importante destacar que, cuando se van a ejecutar los archivos fuente, el primero en ejecutarse debe ser el del servidor. Esto es así porque este tiene que estar preparado para recibir las solicitudes que le hagan los posibles clientes.

Si se ejecutara en primer lugar el código del cliente, estaría haciendo una petición a ningún lugar y el programa fallaría al no encontrar a nadie que atienda sus solicitudes.

Servidor

Para ello en el servidor se define una variable de tipo socket llamado servidor. A continuación, la variable ip se asigna 'localhost' y a port se le asigna el puerto 52816 ya que este suele estar disponible.

A continuación, se enlaza la ip y el puerto y se pone a la escucha al servidor.

A partir de este momento se va a mantener a la escucha para próximas conexiones.

Se encuentran tres opciones, si es la primera iteración de la conexión, se muestra que se está esperando, y se crean dos nuevas variables del tipo de servidor aceptado.

Tras esto se reciben 16 caracteres (cada iteración del bucle) por parte del cliente, en caso de que llegue mensaje, se muestra este envío del cliente, y se le devuelve al cliente, incrementando la variable que indica si es el primer mensaje del cliente en uno. En caso de que no lleguen más mensajes del cliente se muestra que ya no está mandando más mensajes el cliente y se cierra la conexión. Poniendo la variable de primer mensaje a 0.

Cliente

Se crea una variable llamada cliente de tipo socket, otra de tipo ip que se llama 'localhost' y otra que se llama puerto que está definida como 52816. Luego se crea otra variable llamada mensaje que va a contener el mensaje que va a ser enviado al servidor.

Se imprime la ip y el puerto y se conecta por el socket creado a la ip y al puerto.

Manda el mensaje entero al servidor, y este será el que los ira recibiendo de 16 en 16.

A continuación, se hace un bucle mientras que la variable iterador se incrementa hasta llegar a la longitud del menaje, el cliente espera a recibir un mensaje de vuelta por parte del servidor, una vez que llega todo el mensaje de vuelta, e imprimiéndolo por pantalla, el socket del cliente se cierra.

Capturas de las salidas

Servidor

```
starting up on localhost port 52816
Waiting for a connection.
Conection from: 127.0.0.1 55671
Received: b'This is the mess'
Sending data back to the client
Received: b'age. It will be '
Sending data back to the client
Received: b'repeated.'
Sending data back to the client
no more data from: 127.0.0.1 55671
Waiting for a connection.
```

Cliente

```
connect to localhost port 52816
Sending: This is the message. It will be repeated.
Received: b'This is the mess'
Received: b'age. It will be '
Received: b'repeated.'
Closing socket

Process finished with exit code 0
```

ANEXO I

Código del servidor

```
import socket

servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ip = 'localhost'
port = 52816
print("starting up on %s port %d" % (ip, port))
servidor.bind((ip,port))
servidor.listen(1)
i = 0

while True:
    if i == 0:
        print("Waiting for a connection.")
        cliente, direccion = servidor.accept()
        print("Conection from: %s %d" % (direccion[0], direccion[1]))
    respuesta = cliente.recv(16)
    if respuesta:
        print("Received: %s" % respuesta)
        print("Sending data back to the client")
        cliente.send(respuesta)
        i+=1
    else:
        print("no more data from: %s %d" %(direccion[0], direccion[1]))
        cliente.close()
        i=0
```

Código del cliente

```
import socket

cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

ip = 'localhost'
puerto = 52816

mensaje = "This is the message. It will be repeated."

print("connect to %s port %d" %(ip,puerto))
cliente.connect((ip, 52816))
print("Sending: %s" %mensaje)
cliente.send(bytes(mensaje, "utf-8"))

iterador = 0
while iterador < len(mensaje):
    respuesta = cliente.recv(16)
    iterador += len(respuesta)
    print("Received: %s" %respuesta)

print("Closing socket")
cliente.close()
```