**TCP**

**1. ¿Qué llamada sirve para crear el socket? ¿Cuántos parámetros necesita y qué significa cada uno de ellos? ¿Existe alguna diferencia entre las llamadas que efectúan el cliente y el servidor?**

La función socket() sirve para abrir una conexión y devuelve un descriptor al fichero o -1 si se ha producido un error. Los parámetros son los siguientes:

- AF\_INET o AF\_UNIX indica si los clientes pueden estar en otros ordenadores distintos del servidor o van a correr en el mismo ordenador.

- El segundo parámetro indica si el socket está orientado a la conexión (SOCK\_STREAM) o no lo está (SOCK\_DGRAM).

- El tercer parámetro indica el protocolo a emplear. Normalmente se pone 0 dejando al sistema la tarea de la elección del protocolo.

-No existe ninguna diferencia entre las llamadas del cliente y servidor.

**2. ¿Qué llamada hace el servidor para admitir la conexión de un nuevo cliente? ¿Qué devuelve esta llamada? ¿Qué tiene de particular y por qué es necesario este comportamiento?**

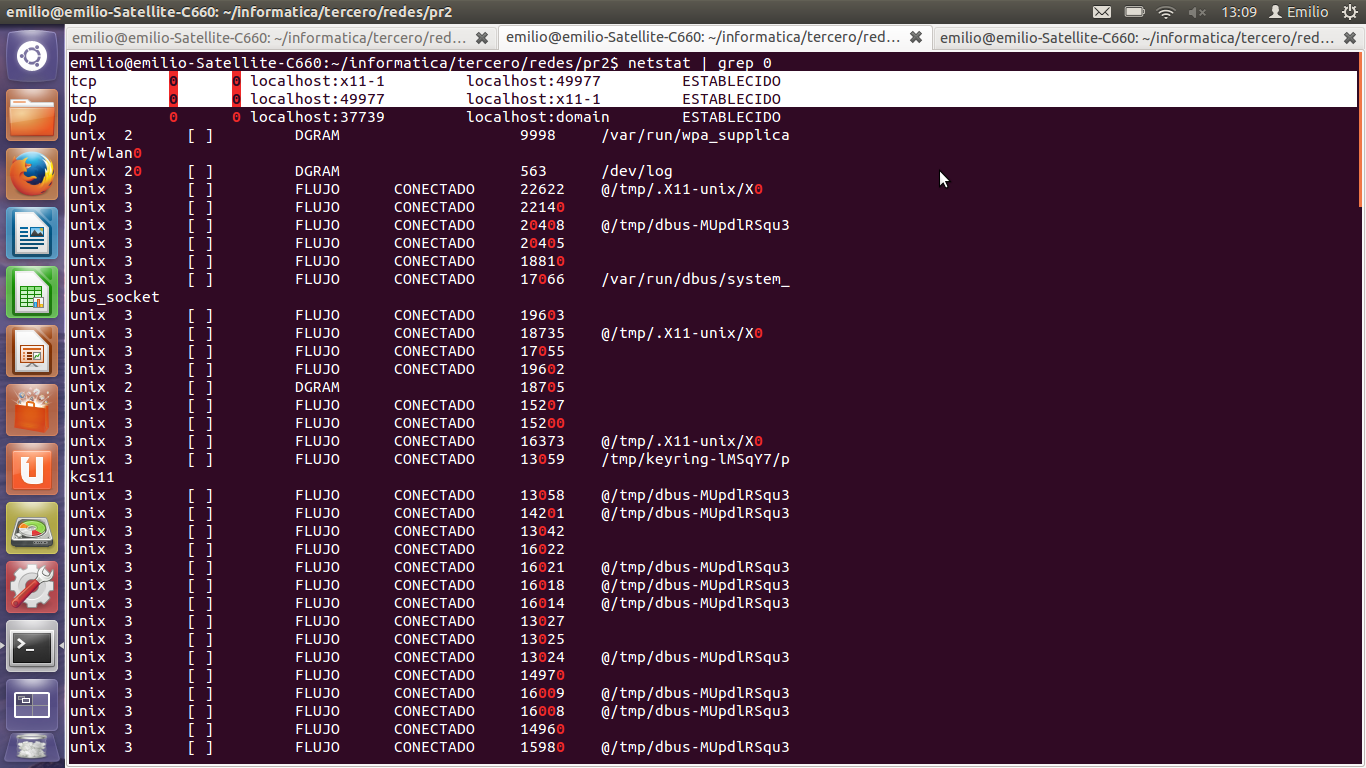
Llamada a función accept(). Esta función devuelve un nuevo descriptor de socket de tipo entero al recibir la conexión del cliente en el puerto configurado. Ésta función tiene de especial que no devolverá valor hasta que se produce una conexión o es interrumpida por una señal.

**3. Tal como está programado el servidor, ¿cuántos clientes concurrentes pueden estar conectados al mismo momento?**

Se puede deducir con la función listen (listen (ID\_socket\_ser, 2)) que el número máximo de conexiones en cola es 2. Sin embargo, el número de clientes concurrentes conectados al mismo tiempo es 1, ya que no hay función que asegure la concurrencia.

**4. Cuando se ejecuten los programas, comprobar con netstat que se han creado los sockets**

**correspondientes, y adjuntar una captura de pantalla.**



**UDP**

## 5. ¿Cuál es la unidad de intercambio de datos entre el cliente y el servidor? ¿Cuántos datos puede transportar, a partir de lo que se observa en el código, cada paquete de datos?

La información se intercambia mediante las funciones recvfrom() y sendto():

- recvfrom() lee un mensaje por un socket udp.

- sendto() sirve para enviar mensajes.

El tamaño de dato a transportar es el número de bytes que ocupa más 1.

**6. Dado que en UDP no se establece conexión entre el cliente y el servidor, ¿dónde define el cliente la dirección y el Puerto del servidor al cuál se quieren enviar los datos?**

Para tal fin se utiliza la función bind(), esta dice al sistema que se desa atender a un determinado servicio. Dicha función tiene la siguiente estructura bind (Descriptor, (struct sockaddr \*)&Direccion, sizeof (Direccion));

La dirección y el puerto del servidor se le pasan con el segundo parámetro struct sockaddr *Dirección:*

Direccion.sin\_family=AF\_INET;

Direccion.sin\_port=htons(numero\_puerto);

Direccion.sin\_addr.s\_addr=INADDR\_ANY;(Direccion));

**7. ¿Qué llamadas usa el cliente para enviar los datos? ¿Qué llamadas usa el servidor para recibir los datos a través de los sockets UDP? ¿Cuántos parámetros tiene cada llamada?**

Para enviar datos usa la llamada sendto (Descriptor, (char \*)&buffer, sizeof(buffer), 0, (struct sockaddr \*)&Cliente, longitudCliente);

Para recibir datos usa la llama recvfrom (Descriptor, (char \*)&buffer, sizeof(buffer), 0, (struct sockaddr \*)&Cliente, &longitudCliente);

Como se puede observar cada llamada tiene 6 parámetros.

**8. ¿Cómo puede saber el servidor quién envía un paquete?**

En el quinto argumento de la función recvfrom(), la función nos devolverá en dicho argumento formado por una estructura los datos del que nos ha enviado el mensaje.

**9. Ejecutar los programas, y comprobar con netstat que se han creado los sockets correspondientes. Adjuntar una captura.**

