The Socket Interface

Chapter 29

Sockets

- A Socket represents a connection between an application and the TCP/IP protocol stack.
- A Socket is defined by an IP address and a port number.
- Sockets are the basis for most application layer network communications.

The Socket API

- Used to Interface an application process and TCP.
- The Socket API supplies hooks called methods or functions that an application programmer can utilize to access the services of TCP.
- These hooks (methods, functions) can be accessed by various programming languages such as C, C++, or Java.
- Each specific language has different syntax, but the underlying semantics are the same.

Native Sockets vs. Socket Libraries

- In UNIX Sockets are part of the Operating System.
- In Windows a Socket Library is used.
- To an application programmer, a Socket Library looks like a Native Socket.
- The library functions become part of the applications and in turn call operating system functions to perform the required operations.
- Java interfaces with the Java Virtual Machine which interfaces with either a Native Socket in UNIX, or a Socket Library in Windows.

Using Server Sockets

- First a server application creates a Server SOCKET.
- Then it uses various methods/functions to:
  - BIND to a port and IP address.
  - LISTEN for clients that wish to connect.
  - ACCEPT a client connection and spin off another socket.
  - CLOSE the SOCKET when the client disconnects.
Creating a Socket

- Definition
  - descriptor = socket(protofamily, type, protocol)

- Example
  - socket1 = socket(PF_INET, SOCK_STREAM, myprotocol)
  - (-1) is returned if there is an error, otherwise an integer is returned and assigned to the socket variable, socket1.

Closing a Socket

- Definition
  - close(socket)

- Example
  - close(socket1)
  - This would close socket1 (note that socket1 is an integer representing the socket descriptor).

Binding a Socket

- Definition
  - bind(socket, localaddr, addrlen)

- Example
  - bind(socket1, srvr_addr, sizeof(srvr_addr))
  - This function binds socket1 to the servers address which includes the port number and the IP address of the server among other things.

Telling the Server to Listen

- Definition
  - listen(socket, queueSize)

- Example
  - listen(socket1, 10)
  - This tells the Server to listen for clients on socket1, and sets the queue size to 10 clients.

Accepting a Client and handing off to another Socket

- Definition
  - newsock = accept(socket, caddress, caddreslen)

- Example
  - socket2 = accept(socket1, client_addr, size(client_addr))
  - The Client is accepted on socket1 and handed off to socket2 for servicing. Note that the client_addr includes both the IP address of the client and the port number used on the client.

Using Client Sockets

- First a Client Application creates a SOCKET.

- Then uses various methods to:
  - BIND to a port and IP address.
  - CONNECT to a Server.
  - CLOSE the Socket when finished with the connection.
Creating a Socket

- Definition
  - descriptor = socket(protofamily, type, protocol)

- Example
  - client_sock = socket(PF_INET, SOCK_STREAM, myprotocol)
  - (-1) is returned if there is an error, otherwise an integer is returned and assigned to the socket variable, client_sock.

Connecting from a Client to a Server

- Definition
  - connect(socket, address, addreslen)

- Example
  - connect(client_socket, server_addr, sizeof(server_addr))
  - Note that client_socket is the socket on the client, server_addr is the server address and includes the IP address of the server and port number on the server.

  - This method connects the Client to the Server.

Both Client and Server Applications

- SEND or WRITE data to one another.
  - Also can use SENDTO, and SENDMSG for connectionless applications.

- RECV or READ data from one another.
  - Also can use RECVFROM and RECVMSG for connectionless applications.

Sending Data Through a Socket

- Definition
  - send(socket, data, length, flags)

- Example
  - send(client_sock, mybuf, strlen(buf), 0)
  - This sends the contents of mybuf (a buffer) to the the socket, client_sock.

Receiving Data from a Socket

- Definition
  - recv(socket, buffer, length, flags)

- Example
  - recv(socket2, &buf, sizeof(buf), 0)
  - This receives data from socket2 which is a socket in the server, and places it in buf. Note that &buf is the location in memory of the buffer.

Threads

- The Main Thread listens for any clients trying to connect.
- When a Client Connects, a Service Thread is spun off to handle the Client.
- Service Threads inherit any Sockets from the Main Thread when they are created.
- When a Service Thread terminates execution it calls the CLOSE method/function for the Socket.