Basics of Socket Programming:

A socket allows one to plug in to the network and communicate with other applications who are plugged in to the same network.

The main types of sockets are:-

- Stream Sockets - Uses TCP as end-to-end protocol. Socket type is SOCK_STREAM.
- Datagram Sockets - Allows the process to use UDP. Socket type is SOCK_DGRAM.

Steps involved in Client and Server's communication

TCP Client's communication

1) Create a socket using socket()
2) Connection is established using connect( )
3) Communicate through send( ) and recv( )
4) Connection is closed using close( )

TCP Server communication

1) Create a socket using socket( )
2) Bind the socket with the port number using bind( )
3) To allow connections to connect to that port use listen( )
4) Repeats until the client close the connection
    a) For accepting new client connection use accept( )
    b) Communicate using send( ) and recv( )
    c) Close the connection using close( )

Difference in the server's use of socket is binding an address to the socket and then using that socket as a way to obtain other sockets connected to clients.

- Creating Socket using socket()
  int socket(int domain, int type, int protocol)
  First parameter specifies the communication domain (here IPv4), second parameter specifies the type of socket (here SOCK_STREAM) and third parameter specifies end-to-end protocol (providing 0 causes system to set default).

- Specifying Address
  The sockaddr structure depends upon the IP structure. Here we are using sockaddr_in for IPv4. This structure contains the port number, address family and Internet address.

- Address Conversion: Using inet_pton (printable to numeric)
  Used to convert IPv4 address in its standard text form to numeric binary form.
  inet_pton(int addressFamily, const char* src, void* dst)

First parameter specifies the address family (here AF_INET), second parameter specifies the address that has to be converted, the third parameter points to a block of memory to keep the result.

- Binding to address: Using bind()
  int bind(int socket, struct sockaddr *localAddress, socklen_t addressSize)

First parameter specifies the file descriptor to be bound, second parameter points to sockaddr structure containing the address to be bound and the third parameter specifies the length.

- Listening to a socket
  int listen(int socket, int Limit)
  The limit parameter specifies the upper bound on the number of incoming connections.

- Accepting connection
  int accept(int socket, struct sockaddr *clientAddress, socklen_t *addressLength)
  This function takes the first connection on the queue of pending connections.

First parameter specifies the socket, second parameter fills in the sockaddr structure pointed to by clientAddress and third parameter specifies the size of the structure.

- Exchanging data
  Communication is done using send( ) or recv( )
  send(int socket, const void *msg, size_t msgLength, int flags)

First parameter is the socket i.e descriptor for the connected socket, second parameter points to a sequence of bytes that has to be sent, third parameter is the number of bytes to be sent. Setting flags to 0 specifies the default behaviour.

recv(int socket, void *rcvBuffer, size_t bufferLength, int flags)

Second parameter points to a buffer (like a character array where received data will be placed), third parameter gives the length of the buffer (Max. bytes received)

Multithreading

A multithreaded program contains two or more tasks that can run parallely. Each part of that program is called a thread. Multithreading can be implemented using POSIX threads (Portable Operating System Interface).

Portability- A program is said to be portable when it can run easily on different OS. Every program has its own set of API(Application Programming Interface) and its interpreter type. (**Shell is the interpreter of LINUX**). POSIX is considered to be the subsystem of UNIX and is used to cover different Unix- like environments for many other operating systems.

Source file #include<pthread.h>