CS558 - Computer System Lab

End Sem Assignment 1

BY: Jayprakash Patidar(194101020) & Mayank sharma (194101030)

- **Introduction**

  A client server communication based on TCP and UDP Sockets. A socket is a combination of IP address and Port number. Sending data from one system to another through a network on any interface follows some standard rules/protocols to communicate on the internet, these rules/protocols are standardised as TCP - Transmission Control Protocol and UDP - User Datagram Protocol.
  Based on application requirements one can use these protocols for sending and receiving data on the internet.

- **Some common socket programming system calls used in our assignment.**

  1. SOCKET system call:
     With the help of this function, the protocol family (TCP, UDP, etc.) is specified.
     Used to create a new socket, returns a file descriptor for the socket or -1 on error.
     It takes three parameters:
     - domain: the protocol family of socket being requested
     - type: the type of socket within that family
     - and the protocol.

       PROTOTYPE:

       int socket(int domain, int type, int protocol)
  2. BIND system call:
     By using bind system call, the server application reserves a port number for itself. Basically it binds IP address with port number and returns a file descriptor, otherwise returns an error if the port number is already being used by another application. It is used only by the server, not by the client.
     PROTOTYPE:
     *int bind(int fd, struct sockaddr *local_addr, socklen_t addr_length)*

3. LISTEN system call :
   Listen system call used after the binding is done i.e. server is associated with the port number and ip address ,then listen make the server to wait for any client to get into steps of start the communication.The listen command is used by the server for specifying the queue length i.e. maximum number of requests a server can handle.
   PROTOTYPE:
   int listen(int fd, int backlog_queue_size)

4. ACCEPT system call:
   Accept system call is a blocking system call that blocks the server until the server gets an incoming request for communication.
   It extracts the first connection request on the queue of pending connections for the listening socket, **sockfd**, creates a new connected socket, and returns a new file descriptor referring to that socket. The newly created socket is not in the listening state. The original socket sockfd is unaffected by this call.
   PROTOTYPE:
   *int accept(int fd, struct sockaddr *remote_host, socklen_t addr_length)*

5. CONNECT system call:
   Connect system call used at client side.its main functionality is that it tries to connect with server on the network.it is blocking call and it block the caller until the connection is made or time limit exceeds.
   PROTOTYPE:
   *int connect(int fd, struct sockaddr *remote_host, socklen_t addr_length)*

6. SEND system call:
   After establishing the connection successfully between client and server, send system call is used to send any data from client to server or vice versa.
   Returns the number of bytes sent or -1 on error.
   PROTOTYPE:
   *int send(int fd, void *buffer, size_t n, int flags)*

7. RECEIVE system call:
   Receive is blocking system call ,it make the system wait for sender to send the data. If any system tries to send the data then other end must be in receiving mode i.e. their must be receive function to receive senders data.
   PROTOTYPE:
   *int receive(int fd, void *buffer, size_t n, int flags)*

8. CLOSE system call:
   This system call used after the successful communication between client and server. This call is used to terminate the connection.
   PROTOTYPE:
   int close(int fildes)

- Algorithm Steps of server program
  1. Initially the program check for command line argument validity.
  2. Now the program converts the port number into integer form from the char array.
  3. Generate the SOCKET system call to get the socket file descriptor for TCP connection.
  4. Generate the BIND system call to bind port number with ip address.
  5. Now it's time to call the LISTEN system call .
  6. Make a call for the ACCEPT system call in while loop because we have to design a concurrent server.
  7. Make a FORK call to create a child process for each client.
  8. According to the problem statement now server wait for client to send the udp port request i.e. Message Type 1.
  9. After the server receives the udp request from the client ,the server generates a random number in the range (1024- 63999) to send it as a response message to the client i.e. message type 2.
  10. After sending the response msg server close the tcp connection.
  11. Now server create the socket for udp connection.
  12. After these server call BIND system call to bind the address for udp connection.
  13. Now server wait for client to send the data, for these server call recvfrom() function to receive udp data.
  14. If data msg is exit then the server program exit from loop and close the udp connection.
  15. If data msg is not exit then the server sends the response msg to the client and again wait for data msg from client in loop.

16. For each client a child process is created so the main process continues in listen state until someone externally forces it to stop.

- Algorithm steps for client program
    1. Initially the program check for command line argument validity.
    2. Now the program converts the port number into integer form from the char array.
    3. After this call the CONNECT system call to make a connection to the server .
    4. After the successful connection with the server client request for udp port number to the server hence the request msg to the server i.e msg type 1.
    5. Now the client wait for a reply from the server so it call RECEIVE system call that blocks the client until the server sends the reply msg .
    6. After receiving the udp port number from the server ,the client closes the tcp connection.
    7. Now client creates the socket for udp connection so call socket system call.
    8. After creating an udp socket client directly sends the msg type 3 to the server on the udp connection .
    9. Now the client waits for the server response i.e msg type 4.
    10. After receiving the msg client program, ask the user to continue to send the data or not as type Y/N.
    11. If user TYPE ‘N’ than udp connection is closed and the program is terminated.
    12. Else again client send the data msg type 3 and wait for a reply in loop.
    13. After ‘N’ choice of user client program terminated .