LAB NO: 2

Aim: Socket Programming in 'C' using TCP -Concurrent Client-Server Programs

```
// Server side program that sends
// a 'hi client' message
// to every client concurrently

//headers providing  necessary functions for network programming
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>

// PORT number used by the server to listen for incoming connections
#define PORT 4444

int main()
{
        // Server socket id or server socket file descriptor
        int sockfd, ret;

        // Server socket address structures (holding the
        //address information for the server)
        struct sockaddr_in serverAddr;

        // Client socket id (holding the socket file
        //descriptor for the connected client)
        int clientSocket;

        // Client socket address structures (holding
        //the address information for the client)
        struct sockaddr_in cliAddr;

        // Stores byte size of server socket address
        socklen_t addr_size;
```

```
// Child process id created by fork()
pid_t childpid;

// Creates a TCP socket id from IPV4 family
sockfd = socket(AF_INET, SOCK_STREAM, 0);

// Error handling if socket id is not valid
if (sockfd < 0)
{
        printf("Error in connection.\n");
        exit(1);
}

printf("Server Socket is created.\n");

// Initializing address structure with NULL
memset(&serverAddr, '\0',sizeof(serverAddr));

// Assign port number and IP address
// to the socket created
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(PORT);

// 127.0.0.1 is a loopback address
serverAddr.sin_addr.s_addr
        = inet_addr("127.0.0.1");

// Associating the socket with the address
//and port specified in serverAddr
ret = bind(sockfd,
                (struct sockaddr*)&serverAddr,
                sizeof(serverAddr));

// Error handling
if (ret < 0) {
        printf("Error in binding.\n");
        exit(1);
}
```

```
// Server listening for connections (upto 10)
if (listen(sockfd, 10) == 0) {
        printf("Listening...\n\n");
}

int cnt = 0;
while (1) {

        // Accept clients connection and
        // store their information in cliAddr
        clientSocket = accept(
                sockfd, (struct sockaddr*)&cliAddr,
                &addr_size);

        // Error handling
        if (clientSocket < 0) {
                exit(1);
        }

        // Displaying information(IP address and port number)
        // of connected client
        printf("Connection accepted from %s:%d\n",
                inet_ntoa(cliAddr.sin_addr),
                ntohs(cliAddr.sin_port));

        // Print number of clients
        // connected till now
        printf("Clients connected: %d\n\n",
                ++cnt);
```

```
                // Creates a child process
                if ((childpid = fork()) == 0) {

                        // Closing the server socket id
                        close(sockfd);

                        // Send a confirmation message
                        // to the client
                        send(clientSocket, "hi client",
                                strlen("hi client"), 0);
                }
        }

        // Close the client socket id
        close(clientSocket);
        return 0;


// Accept connection request from client in cliAddr
// socket structure
clientSocket = accept(sockfd, (struct sockaddr*)&cliAddr, &addr_size);

// Make a child process by fork() and check if child
// process is created successfully
if ((childpid = fork()) == 0)
{
        // Send a confirmation message to the client for
        // successful connection
        send(clientSocket, "hi client", strlen("hi client"),0);
}
}
```

```c
// Client Side program to test
// the TCP server that returns
// a 'hi client' message

#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>

// PORT number
#define PORT 4444

int main()
{
        // Socket id(client socket file descriptor)
        int clientSocket, ret;

        // Client socket structure ( holding address information for the client & server)
        struct sockaddr_in cliAddr;

        struct sockaddr_in serverAddr;

        // char array to store incoming message received from the server
        char buffer[1024];

        // Creating socket id
        clientSocket = socket(AF_INET,SOCK_STREAM, 0);
```

```
if (clientSocket < 0) {
        printf("Error in connection.\n");
        exit(1);
}
printf("Client Socket is created.\n");

// Initializing socket structure with NULL
memset(&cliAddr, '\0', sizeof(cliAddr));

// Initializing buffer array with NULL
memset(buffer, '\0', sizeof(buffer));

// Assigning port number and IP address
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(PORT);

// 127.0.0.1 is Loopback IP
serverAddr.sin_addr.s_addr
        = inet_addr("127.0.0.1");

// connect() to connect to the server using the address and port specified in serverAddr.
ret = connect(clientSocket,
                        (struct sockaddr*)&serverAddr,
                        sizeof(serverAddr));

if (ret < 0) {
        printf("Error in connection.\n");
        exit(1);
}

printf("Connected to Server.\n");
```

```
         //infinite loop to continuously receive messages from the server
         while (1) {

                 // recv() receives the message
                 // from server and stores in buffer
                 if (recv(clientSocket, buffer, 1024, 0)
                         < 0) {
                         printf("Error in receiving data.\n");
                 }

                 // Printing the message on screen if a message is successfully received
                 else {
                         printf("Server: %s\n", buffer);
                         bzero(buffer, sizeof(buffer));
                 }
         }

         return 0;
 }
```