EXPERIMENT: 29

CREATING THE APPLICATIONS USING TCP ECHO SERVER AND CLIENT IN JAVA/C

Aim: To create Applications using TCP ECHO SERVER and CLIENT.

Algorithm :

SERVER:

STEP1:Start

STEP2: Declare the variables for the socket

STEP3: Specify the family, protocol, IP address and port number STEP4:

Create a a socket using socket() function

STEP 5: Bind the IP address and Port number

STEP6: Listen and accept the client's request for the connection

STEP7: Read the client's message STEP8:

Display the client's message STEP 9: Close

the socket

STEP10: Stop

CLIENT:

STEP1:Start

STEP2: Declare the variables for the socket

STEP3: Specify the family, protocol IPaddress and port number STEP4:

Create a socket using socket() function

STEP5: Call the connect() function STEP6:

Read the put message

STEP7: Send the input message to the server

STEP 8: Display the server's echo

STEP 9: Close the socket

STEP 10: Stop

Procedure:

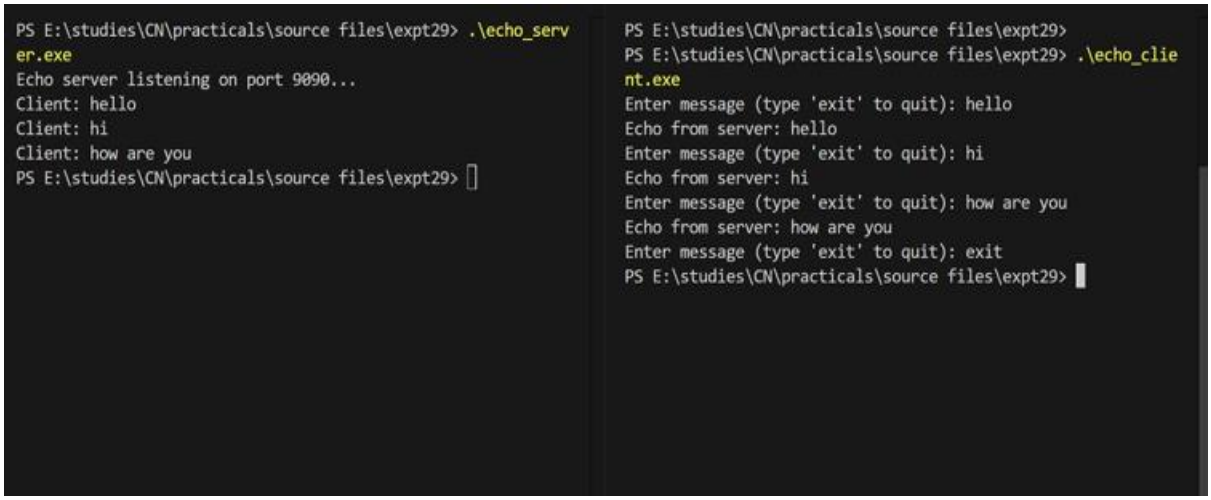
TCP Echo Server-side implementation:

1. Create a TCP socket using the `socket()` function with the `AF_INET` address family and `SOCK_STREAM` socket type.
2. Set socket options using the `setsockopt()` function to allow reuse of the address and port.
3. Bind the socket to a specific IP address and port using the `bind()` function.
4. Listen for incoming connections using the `listen()` function.
5. Accept a client connection using the `accept()` function, which returns a new socket descriptor for the accepted connection.
6. Receive data from the client using the `recv()` function on the accepted socket descriptor.
7. Process the received data if necessary.
8. Optionally, send a response back to the client using the `send()` function on the accepted socket descriptor.
9. Close the accepted socket descriptor using the `close()` function.
10. Close the server socket descriptor using the `close()` function.

TCP Echo Client-side implementation:

1. Create a TCP socket using the `socket()` function with the `AF_INET` address family and `SOCK_STREAM` socket type.
2. Set the server address and port in a `struct sockaddr_in` structure.
3. Connect to the server using the `connect()` function with the server socket descriptor and the server address structure.
4. Send data from the client to the server using the `send()` function on the connected socket descriptor.
5. Receive the response from the server using the `recv()` function on the connected socket descriptor.
6. Process and display the received data as needed.
7. Close the connected socket descriptor using the `close()` function.

The echo server simply returns back the received data, allowing the client to see the echoed message. Remember to include the necessary header files (`<stdio.h>`, `<stdlib.h>`, `<string.h>`, `<sys/socket.h>`, `<netinet/in.h>`, etc.) and handle errors appropriately in the code.



Result: Thus the Applications using TCP ECHO SERVER AND CLIENT is created successfully.