Computer Networks

Name : M. Sai Saranya

Regno: 22BAI1471

Course Title : Computer Networks

Course code : BCSE308P

Slot : L45-46

Faculty : Dr Neelanarayanan V

S.No	Experiment Name	Date	Page No.	Marks
2.	**Client-Server Application Echo**	**17-01-2024**		

Experiment No. 2

Experiment Name: Client-Server Application (Echo client-server)

Date: 17-1-2024

Problem Statement

Design a simple client-server application named Echo client-server using c program and execute in Linux.

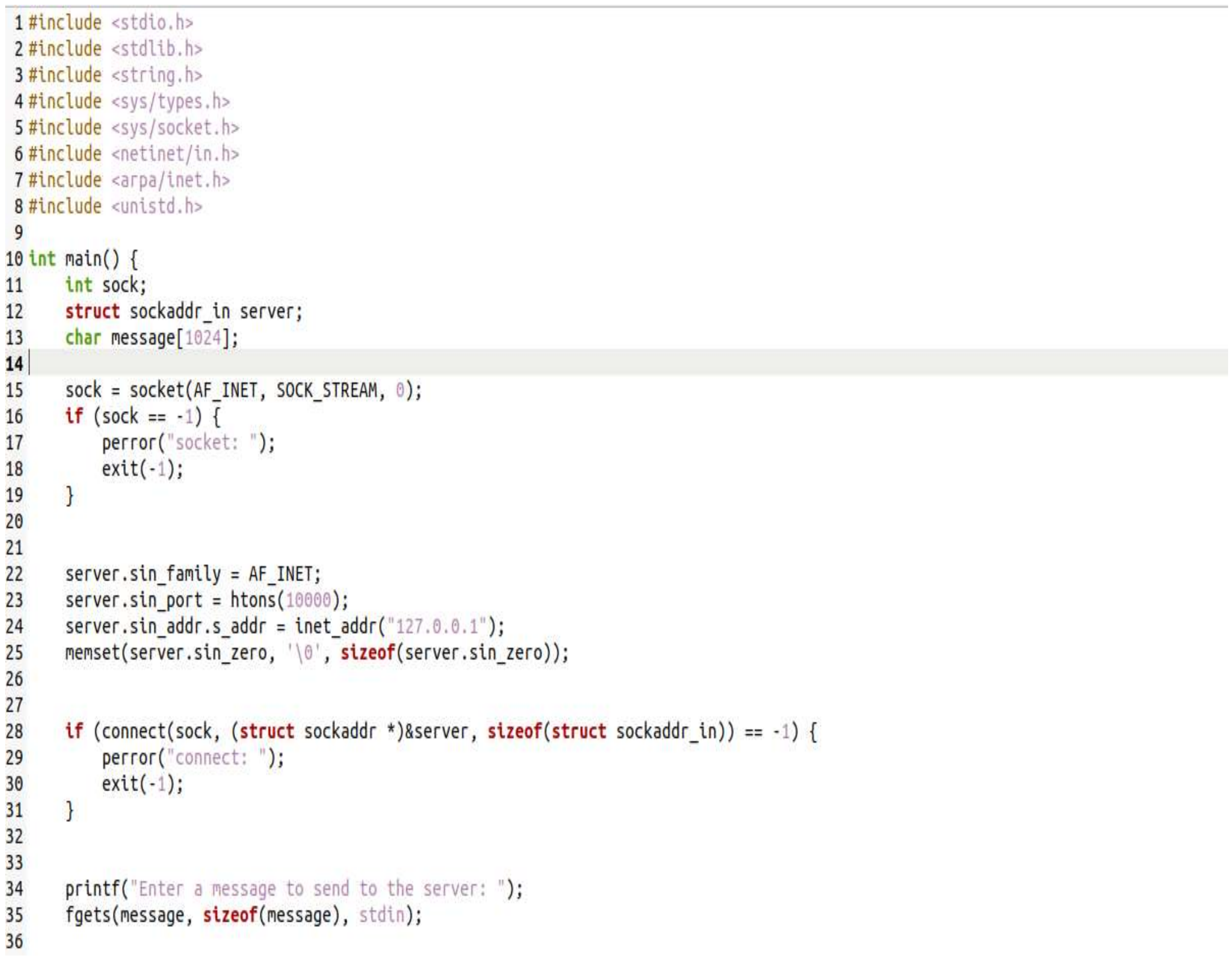
Aim

To write a c program for echo client-server application (tcp protocol) and execute in Linux environment.

Algorithm or Procedure

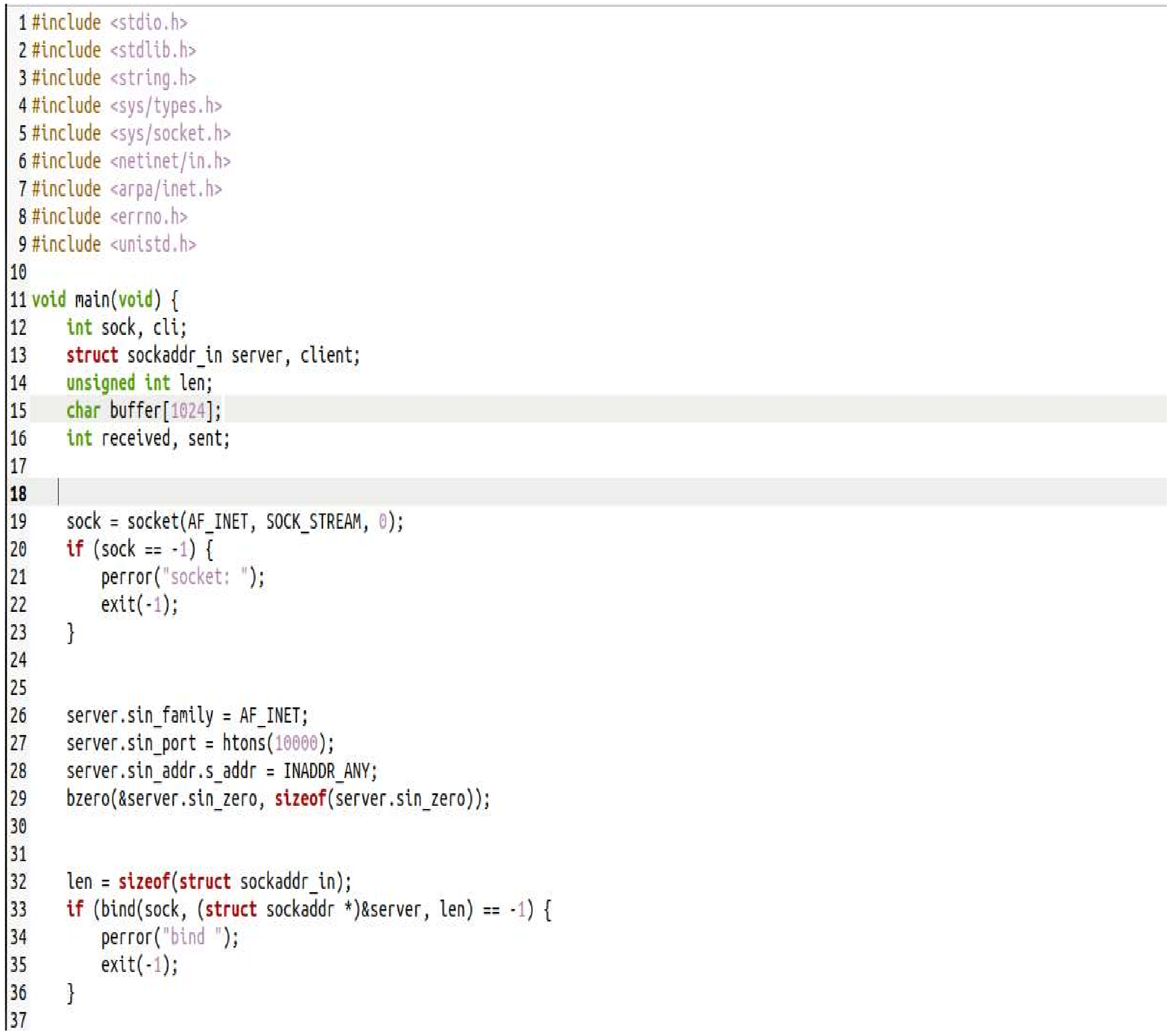
1. Writing client and server files separately using socket programming
2. Execute client and server files separately or simultaneously in two parallel linux terminal windows.
3. Giving the message that is to be return back by server through client.
4. Obtaining desired output

Client Program



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

int main() {
    int sock;
    struct sockaddr_in server;
    char message[1024];

    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1) {
        perror("socket: ");
        exit(-1);
    }


    server.sin_family = AF_INET;
    server.sin_port = htons(10000);
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    memset(server.sin_zero, '\0', sizeof(server.sin_zero));


    if (connect(sock, (struct sockaddr *)&server, sizeof(struct sockaddr_in)) == -1) {
        perror("connect: ");
        exit(-1);
    }


    printf("Enter a message to send to the server: ");
    fgets(message, sizeof(message), stdin);


    printf("Enter a message to send to the server: ");
    fgets(message, sizeof(message), stdin);


    if (send(sock, message, strlen(message), 0) == -1) {
        perror("send: ");
        exit(-1);
    }


    if (recv(sock, message, sizeof(message), 0) == -1) {
        perror("recv: ");
        exit(-1);
    }


    printf("Received from server: %s", message);


    close(sock);

    return 0;
}
```

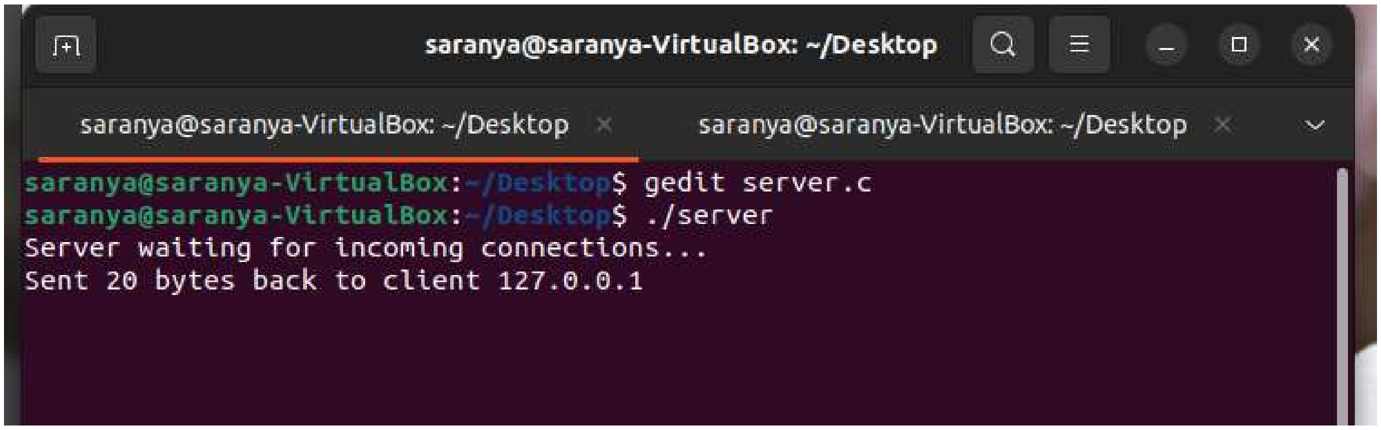
Server Program



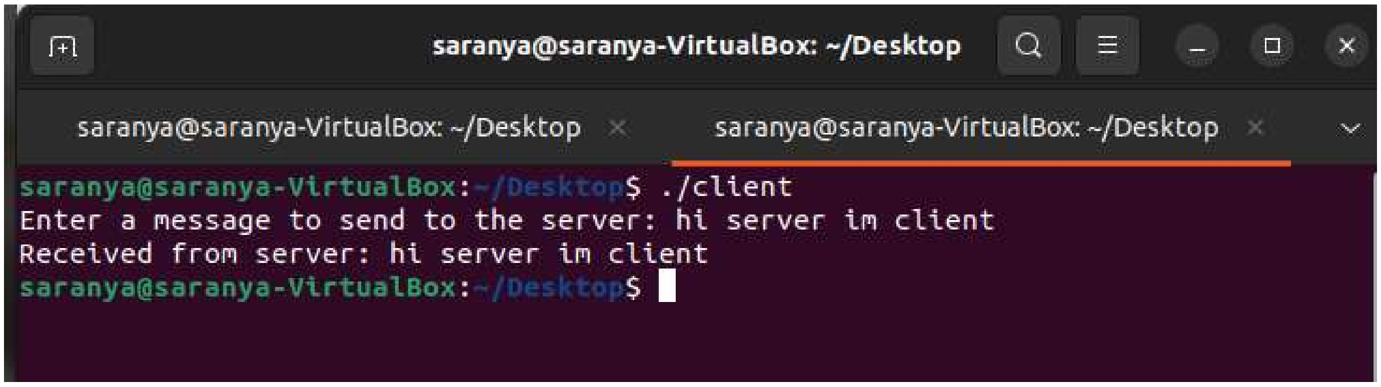
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>
#include <unistd.h>

void main(void) {
    int sock, cli;
    struct sockaddr_in server, client;
    unsigned int len;
    char buffer[1024];
    int received, sent;


    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1) {
        perror("socket: ");
        exit(-1);
    }


    server.sin_family = AF_INET;
    server.sin_port = htons(10000);
    server.sin_addr.s_addr = INADDR_ANY;
    bzero(&server.sin_zero, sizeof(server.sin_zero));


    len = sizeof(struct sockaddr_in);
    if (bind(sock, (struct sockaddr *)&server, len) == -1) {
        perror("bind ");
        exit(-1);
    }

    if (listen(sock, 5) == -1) {
        perror("listen ");
        exit(-1);
    }

    printf("Server waiting for incoming connections...\n");


    while (1) {
        if ((cli = accept(sock, (struct sockaddr *)&client, &len)) == -1) {
            perror("accept ");
            exit(-1);
        }


        received = recv(cli, buffer, sizeof(buffer), 0);
        if (received <= 0) {
            if (received == 0) {
                printf("Connection closed by client\n");
            } else {
                perror("recv ");
            }
            close(cli);
            continue;
        }


        sent = send(cli, buffer, received, 0);
        if (sent == -1) {
            perror("send ");
        } else {
            printf("Sent %d bytes back to client %s\n", sent, inet_ntoa(client.sin_addr));
        }

        close(cli);
    }
}
```

Linux terminal : Output

When a message is entered in the command prompt, the same message is reflected or sent back to the client by the server thus making an echo. Server additionally prints the number of bytes sent to the client and the ip address of the client.


saranya@saranya-VirtualBox: ~/Desktop
saranya@saranya-VirtualBox: ~/Desktop
saranya@saranya-VirtualBox: ~/Desktop

```
saranya@saranya-VirtualBox:~/Desktop$ gedit server.c
saranya@saranya-VirtualBox:~/Desktop$ ./server
Server waiting for incoming connections...
Sent 20 bytes back to client 127.0.0.1
```


saranya@saranya-VirtualBox: ~/Desktop
saranya@saranya-VirtualBox: ~/Desktop
saranya@saranya-VirtualBox: ~/Desktop

```
saranya@saranya-VirtualBox:~/Desktop$ ./client
Enter a message to send to the server: hi server im client
Received from server: hi server im client
saranya@saranya-VirtualBox:~/Desktop$
```