NAME

MOHAMMAD SHAAD

REGISTRATION NUMBER

21BCE1542

CLASS

COMPUTER NETWORKS

FACULTY

PUNITHA K MA'AM

LAB

EXERCISE 6

AIM

TO IMPLEMENT THE SOCKET PROGRAMMING USING **TCP**

PROCEDURE

1. Open two terminals (one for server and another one for client)
2. Execute the server socket program
3. Execute the client program

CODES

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <time.h>

#define BUF_SIZE 1024

void error_handler(char *msg) {
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[]) {

    if (argc != 2) {
```

```
        fprintf(stderr, "Usage: %s <port>\n", argv[0]);
        exit(1);
    }

    int server_sock, client_sock;
    struct sockaddr_in server_addr, client_addr;

    // Create socket
    server_sock = socket(AF_INET, SOCK_STREAM, 0);
    if (server_sock == -1) {
        error_handler("Failed to create socket");
    }

    // Bind socket to address and port
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(atoi(argv[1]));

    if (bind(server_sock, (struct sockaddr *)&server_addr,
sizeof(server_addr)) == -1) {
        error_handler("Failed to bind socket");
    }

    // Listen for connections
    if (listen(server_sock, 5) == -1) {
        error_handler("Failed to listen for connections");
    }

    printf("Server started. Waiting for connections...\n");
```

```
    socklen_t client_addr_size = sizeof(client_addr);

    // Accept incoming connections
    client_sock = accept(server_sock, (struct sockaddr
*)&client_addr, &client_addr_size);
    if (client_sock == -1) {
        error_handler("Failed to accept connection");
    }

    printf("Client connected: %s:%d\n",
inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));

    char buf[BUF_SIZE];
    int read_size;

    while ((read_size = recv(client_sock, buf, BUF_SIZE, 0)) >
0) {

        // Get current time
        time_t now = time(0);
        struct tm *local_time = localtime(&now);
        char timestamp[20];
        strftime(timestamp, sizeof(timestamp), "%Y-%m-%d
%H:%M:%S", local_time);

        // Add timestamp to message
        char message[BUF_SIZE + 38]; // Maximum possible
string length is 19 + BUF_SIZE - 1 + 19
        sprintf(message, "[%s] %s", timestamp, buf);

        printf("%s", message);
```

```
        // Send message back to client
        if (send(client_sock, message, strlen(message), 0) ==
-1) {
            error_handler("Failed to send message");
        }

        memset(buf, 0, sizeof(buf));
    }

    if (read_size == 0) {
        puts("Client disconnected");
    } else {
        error_handler("Failed to receive data");
    }

    close(client_sock);
    close(server_sock);

    return 0;
}
```

client.c

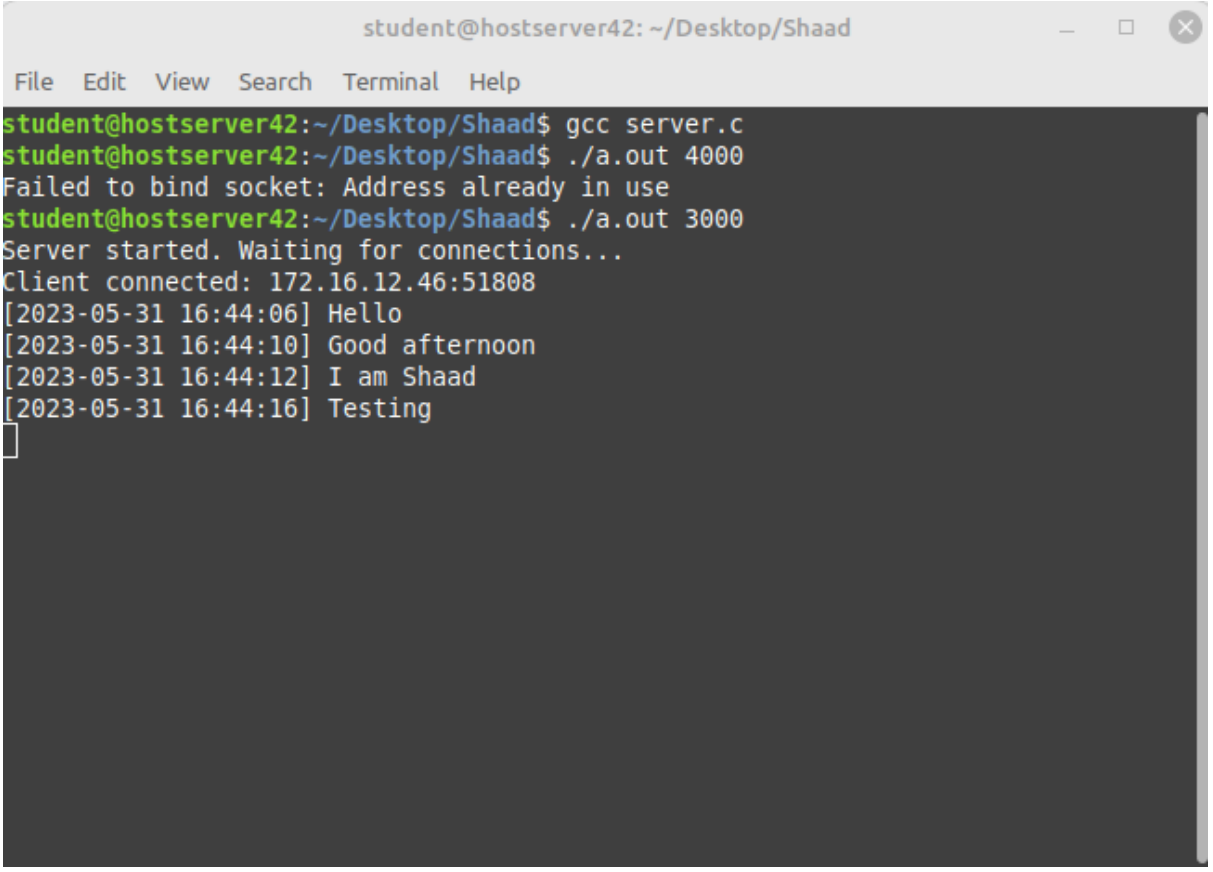
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
```

```
#define BUF_SIZE 1024

void error_handler(char *msg) {
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[]) {

    if (argc != 3) {
        fprintf(stderr, "Usage: %s <server_ip> <port>\n",
argv[0]);
        exit(1);
    }

    int sock;
    struct sockaddr_in server_addr;

    // Create socket
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1) {
        error_handler("Failed to create socket");
    }

    // Configure server address
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(argv[1]);
    server_addr.sin_port = htons(atoi(argv[2]));

    // Connect to server
```

```
    if (connect(sock, (struct sockaddr *)&server_addr,
sizeof(server_addr)) == -1) {
        error_handler("Failed to connect to server");
    }

    printf("Connected to server: %s:%d\n", argv[1],
atoi(argv[2]));

    char message[BUF_SIZE];
    int read_size;

    while (1) {
        printf("Enter message: ");
        fgets(message, BUF_SIZE, stdin);

        // Send message to server
        if (send(sock, message, strlen(message), 0) == -1) {
            error_handler("Failed to send message");
        }

        // Receive response from server
        memset(message, 0, sizeof(message));
        read_size = recv(sock, message, BUF_SIZE, 0);
        if (read_size == -1) {
            error_handler("Failed to receive response");
        } else if (read_size == 0) {
            puts("Server disconnected");
            break;
        }

        printf("Server response: %s", message);
```

```
    }

    close(sock);

    return 0;
}
```

OUTPUT



student@hostserver42: ~/Desktop/Shaad

File Edit View Search Terminal Help

```
student@hostserver42:~/Desktop/Shaad$ gcc client.c
student@hostserver42:~/Desktop/Shaad$ ./a.out 172.16.12.46 3000
Connected to server: 172.16.12.46:3000
Enter message: Hello
Server response: [2023-05-31 16:44:06] Hello
Enter message: Good afternoon
Server response: [2023-05-31 16:44:10] Good afternoon
Enter message: I am Shaad
Server response: [2023-05-31 16:44:12] I am Shaad
Enter message: Testing
Server response: [2023-05-31 16:44:16] Testing
Enter message:
```