NITHIN S
221IT085

Write a Socket Program to implement menu driven calculator where operands and operators are sent by the client and the server replies with the result.

Server Code

server.c

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

double calculate(double operand1, double operand2, char operator)
{
   switch (operator) {
      case '+':
         return operand1 + operand2;
      case '-':
         return operand1 - operand2;
      case '*':
         return operand1 * operand2;
      case '/':
         if (operand2 != 0)
            return operand1 / operand2;
         else
            return -1; // Division by zero error
```

```
        default:
            return -1; // Invalid operator
    }
}

int main() {
    int server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket == -1) {
        perror("Error creating server socket");
        exit(1);
    }
    printf("Waiting for connection...\n");

    struct sockaddr_in servaddr;
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(8080);

    if (bind(server_socket, (struct sockaddr *)&servaddr,
sizeof(servaddr)) == -1) {
        perror("Error binding server socket");
        close(server_socket);
        exit(1);
    }

    if (listen(server_socket, 1) == -1) {
        perror("Error listening for connections");
        close(server_socket);
        exit(1);
    }

    int clifd = accept(server_socket, NULL, NULL);
    if (clifd == -1) {
```

```
        perror("Error accepting client connection");
        close(server_socket);
        exit(1);
    }
    printf("Client Connected.\n");

    double operand1, operand2, result;
    char operator;
    while (1) {
        int valread = read(clifd, &operand1, sizeof(double));
        if (valread <= 0)
            break;

        valread = read(clifd, &operand2, sizeof(double));
        if (valread <= 0)
            break;

        valread = read(clifd, &operator, sizeof(char));
        if (valread <= 0)
            break;

        result = calculate(operand1, operand2, operator);
        write(clifd, &result, sizeof(double));
    }
    close(server_socket);
    close(clifd);
    return 0;
}
```

EXPLANATION

This is a simple server program for handling arithmetic operations requested by a client over a network socket. Below is a brief explanation of what the code does:

1. **Socket Initialization**:
   - The program begins by creating a server socket using the `socket` function. It checks for errors during socket creation.

2. **Server Address Configuration**:
   - It sets up the server's address configuration by initializing a `sockaddr_in` structure (`servaddr`) with the desired family (AF_INET for IPv4), port (8080 in this case), and other necessary parameters.

3. **Binding the Server**:
   - The program binds the server socket to the specified address and port using the `bind` function. It checks for errors during binding and handles them gracefully.

4. **Listening for Connections**:
   - Using the `listen` function, the server socket is set to listen for incoming client connections. It allows only one pending connection (parameter set to 1).

5. **Accepting Client Connection**:
   - The server waits for a client to connect using the `accept` function. When a client connects, it accepts the connection and creates a new socket (`clifd`) dedicated to that client.

6. **Handling Client Requests**:
   - The server enters a loop to handle client requests indefinitely.
   - It reads operands and an operator from the client using the `read` function. It checks for errors or the client disconnecting (indicated by a return value less than or equal to 0) and breaks out of the loop if necessary.
   - The server then calculates the result of the arithmetic operation using the `calculate` function and sends it back to the client using the `write` function.

7. **Closing Sockets**:
   - After the client disconnects or an error occurs, the server closes both the server socket and the client socket (`clifd`) to release resources.

8. **`calculate` Function**:
   - This function takes two operands (`operand1` and `operand2`) and an operator (`operator`) as arguments.
   - It performs a basic arithmetic operation based on the operator and returns the result.

- The function handles addition, subtraction, multiplication, and division (with checks for division by zero and invalid operators).

In summary, this server program listens for client connections, performs basic arithmetic operations based on client requests, and sends the results back to clients. It incorporates error handling and gracefully closes sockets when necessary. The server uses standard socket programming techniques to communicate with clients over a network.

CLIENT CODE

client.c

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

int main() {
    int cli_sock = socket(AF_INET, SOCK_STREAM, 0);
    if (cli_sock == -1) {
        perror("Error creating client socket");
        exit(1);
    }

    struct sockaddr_in servaddr;
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(8080);
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    if (connect(cli_sock, (struct sockaddr *)&servaddr,
sizeof(servaddr)) == -1) {
        perror("Error connecting to the server");
        close(cli_sock);
        exit(1);
```

```
}

while (1) {
    printf("Menu:\n");
    printf("1. Addition\n");
    printf("2. Subtraction\n");
    printf("3. Multiplication\n");
    printf("4. Division\n");
    printf("5. Quit\n");
    printf("Enter your choice: ");
    int choice;
    scanf("%d", &choice);

    if (choice == 5) {
        break;
    }

    double operand1, operand2, result;
    char operator;

    printf("Enter operand 1: ");
    scanf("%lf", &operand1);

    printf("Enter operand 2: ");
    scanf("%lf", &operand2);

    switch (choice) {
        case 1:
            operator = '+';
            break;
        case 2:
            operator = '-';
            break;
        case 3:
            operator = '*';
            break;
        case 4:
            operator = '/';
            break;
        default:
            printf("Invalid choice\n");
```

```
            continue;
        }

        write(cli_sock, &operand1, sizeof(double));
        write(cli_sock, &operand2, sizeof(double));
        write(cli_sock, &operator, sizeof(char));

        read(cli_sock, &result, sizeof(double));

        printf("Result: %lf\n", result);
    }

    close(cli_sock);
    return 0;
}
```

EXPLANATION

This is a simple client program that communicates with a server over a network socket. Here's a brief explanation of what the code does:

1. ****Socket Initialization**:**
   - The program starts by creating a client socket using the `socket` function. It checks for any errors during socket creation.

2. ****Server Address Configuration****:
   - It configures the server's address by setting up a `sockaddr_in` structure (`servaddr`) with the server's IP address (in this case, "127.0.0.1" for the local machine) and port 8080.

3. ****Connection to Server**:**
   - It establishes a connection to the server using the `connect` function. If the connection fails, it displays an error message and exits the program.

4. ****User Interaction**:**
   - The client enters a loop, providing a menu of arithmetic operations to the user (addition, subtraction, multiplication, division, and quitting the program).
   - It reads the user's choice and operands (operand1 and operand2) from the console.

5. ****Sending Data to Server****:
   - Based on the user's choice, the client sends the operator and operands to the server using the `write` function. The data is sent in binary format.

6. ****Receiving and Displaying Results****:

- After sending the data to the server, the client waits for the server to perform the calculation and send back the result. It reads the result from the server using the `read` function and displays it to the user.

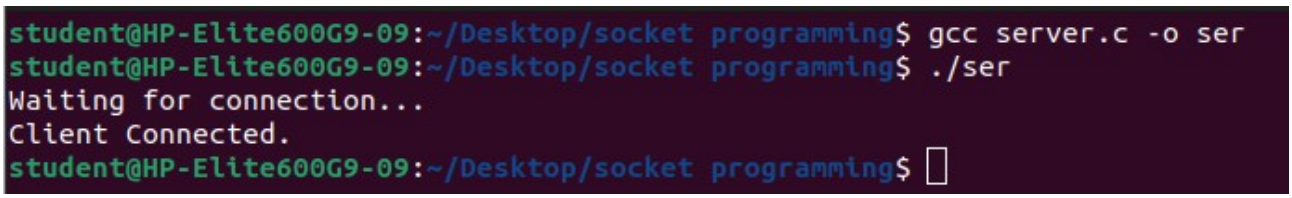
7. **Loop Continuation and Termination**:
   - The client continues to provide the menu and process user requests until the user chooses to quit (option 5). At that point, it breaks out of the loop and proceeds to close the client socket.

8. **Socket Closure**:
   - Finally, the client program closes the client socket using the `close` function before exiting.

In summary, this client program connects to a server, interacts with the user to request arithmetic operations and operands, sends the data to the server, receives and displays the results, and repeats the process until the user chooses to quit. It serves as a basic client for a server that performs simple arithmetic calculations.

OUTPUT

SERVER:



CLIENT:

```
student@HP-Elite600G9-09:~/Desktop/socket programming$ gcc client.c -o cli
student@HP-Elite600G9-09:~/Desktop/socket programming$ ./cli
Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Quit
Enter your choice: 3
Enter operand 1: 2
Enter operand 2: 8.8
Result: 17.600000
Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Quit
Enter your choice: 4
Enter operand 1: 5
Enter operand 2: 8
Result: 0.625000
Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Quit
Enter your choice: 1
Enter operand 1: 4
Enter operand 2: 7.9
Result: 11.900000
Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Quit
Enter your choice: 2
Enter operand 1: 4
Enter operand 2: 6
Result: -2.000000
Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Quit
Enter your choice: 5
student@HP-Elite600G9-09:~/Desktop/socket programming$
```