CS6111 - COMPUTER NETWORKS LAB 2

SOCKET PROGRAMMING – SPOT EXERCISE

Name : Ajitesh M

Reg. No : 2019103503

Aim :

To build a calculator application using socket programming, handling two clients.

Code :

calcServer.c

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

#define PORT 3503
#define MAX 1024

int client_sockets[3];

int calculate(int a, int b, char op)
{
    switch (op)
    {
    case '+':
        return a + b;
    case '-':
        return a - b;
    case '*':
        return a * b;
    case '/':
        return a / b;
    }
}

void *calculator(void *cl_socket)
{
    //pthread_detach(pthread_self());
    int clientSocket = *((int *)cl_socket);
    char buffer[MAX];
    int idx = 0;
    for (int i = 0; i <= 2; i++)
```

```
    {
        if (client_sockets[i] == clientSocket)
        {
            idx = i;
            break;
        }
    }
    while (1)
    {
        bzero(buffer, MAX);
        int len = read(clientSocket, buffer, MAX);
        if (strncmp(buffer, "X", 1) == 0)
        {
            close(client_sockets[idx]);
            client_sockets[idx] = 0;
            break;
        }
        buffer[len] = '\0';
        int op1 = atoi(buffer);

        bzero(buffer, MAX);
        len = read(clientSocket, buffer, MAX);
        buffer[len] = '\0';
        int op2 = atoi(buffer);

        bzero(buffer, MAX);
        len = read(clientSocket, buffer, MAX);
        char operator= buffer[0];

        int result = calculate(op1, op2, operator);

        bzero(buffer, MAX);
        sprintf(buffer, "Result : %d", result);
        write(clientSocket, buffer, MAX);
    }
    printf("[+] Client %d disconnected.\n", idx);
    //pthread_exit(NULL);
}

int main()
{
    pthread_t client_threads[2];

    int serverSocket;
    struct sockaddr_in servAddr;

    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket < 0)
    {
        printf("[-] Socket creation failed.\n");
        exit(1);
    }
    printf("[+] Socket created.\n");
```

```
    int opt = 1;
    if (setsockopt(serverSocket, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)) < 0)
    {
        printf("[-] Unable to set socket options.\n");
        exit(1);
    }

    for (int i = 0; i <= 2; i++)
        client_sockets[i] = 0;

    bzero(&servAddr, sizeof(servAddr));
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(PORT);
    servAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    if (bind(serverSocket, (struct sockaddr *)&servAddr, sizeof(servAddr)) < 0)
    {
        printf("[-] Socket unable to bind.\n");
        exit(1);
    }

    if (listen(serverSocket, 5) < 0)
    {
        printf("[-] Socket unable to listen.\n");
        exit(1);
    }
    printf("[+] Server listening at port %d\n", PORT);

    int idx = 3;
    while (1)
    {
        for (int i = 0; i <= 2; i++)
        {
            if (client_sockets[i] == 0)
            {
                idx = i;
                break;
            }
        }

        client_sockets[idx] = accept(serverSocket, NULL, NULL);
        if (client_sockets[idx] < 0)
        {
            printf("[-] Unable to accept client request.\n");
            exit(1);
        }

        pthread_create(&client_threads[idx], NULL, calculator, (void
*)&client_sockets[idx]);
    }
    for (int i = 0; i < 2; i++)
        pthread_join(client_threads[i], NULL);
```

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

Explanation :

In order to handle multiple client connections, threads are used. Unlike creating a child process for each client using fork(), threads are more resource efficient and can be created to execute a particular sub-routine. Here, one thread for each client is created to execute the calculator() routine which receives the operands and operators for performing the calculation and sends the result to the appropriate client.

calcClient.c

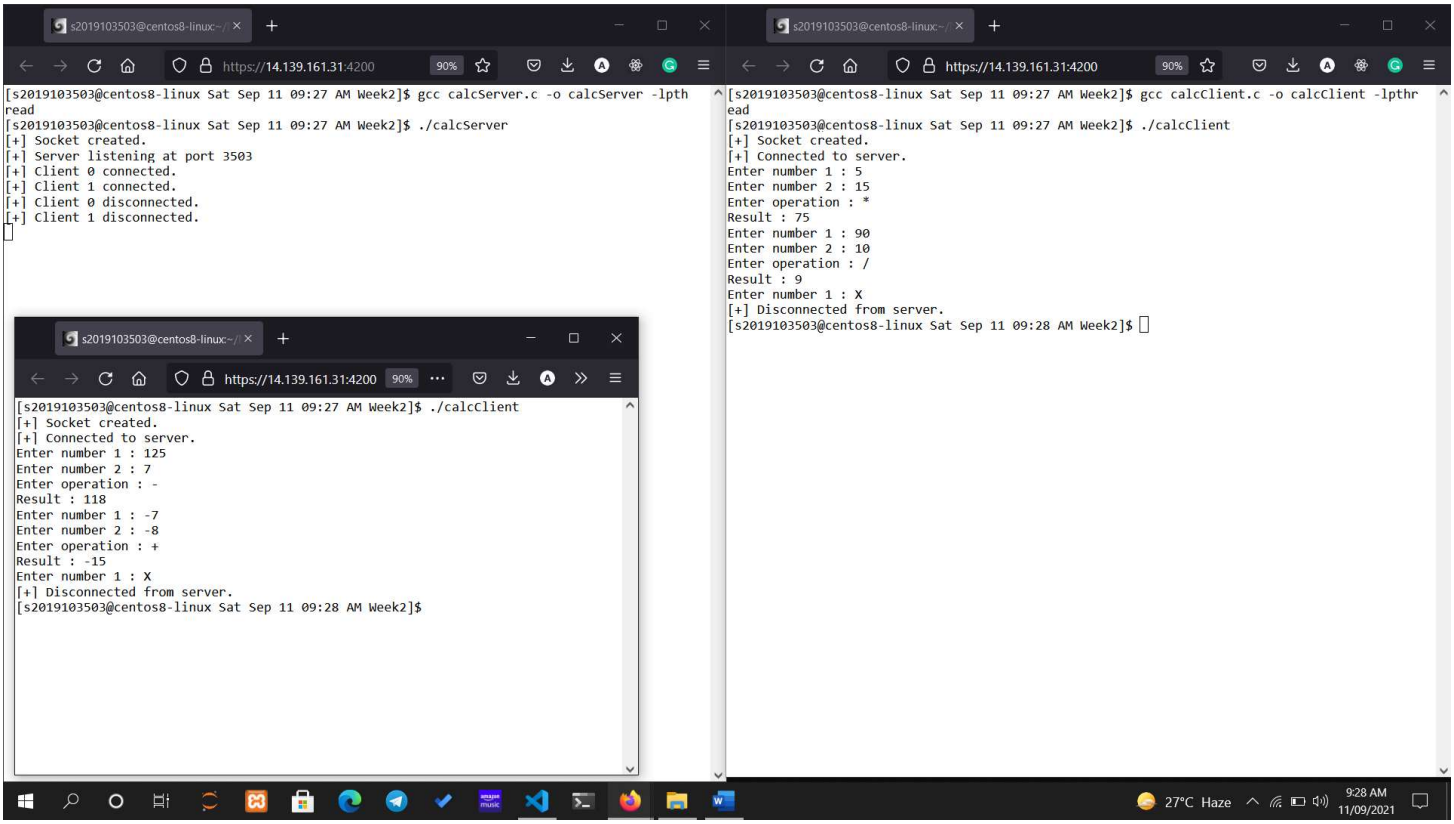
```
--------------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <pthread.h>

#define PORT 3503
#define MAX 1024

int main()
{
    int clientSocket;
    struct sockaddr_in servAddr;

    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket < 0)
    {
        printf("[-] Socket creation failed.\n");
        exit(1);
    }
    printf("[+] Socket created.\n");

    bzero(&servAddr, sizeof(servAddr));
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(PORT);
    servAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    if (connect(clientSocket, (struct sockaddr *)&servAddr, sizeof(servAddr)) <
0)
    {
        printf("[-] Unable to connect to server.\n");
        exit(1);
    }
```

```
    printf("[+] Connected to server.\n");

    while (1)
    {
        char buffer[MAX];
        int n = 0;
        bzero(buffer, MAX);
        printf("Enter number 1 : ");
        while ((buffer[n++] = getchar()) != '\n')
            ;
        write(clientSocket, buffer, MAX);
        if (strncmp(buffer, "X", 1) == 0)
        {
            printf("[+] Disconnected from server.\n");
            break;
        }
        n = 0;
        bzero(buffer, MAX);
        printf("Enter number 2 : ");
        while ((buffer[n++] = getchar()) != '\n')
            ;
        write(clientSocket, buffer, MAX);
        n = 0;
        bzero(buffer, MAX);
        printf("Enter operation : ");
        while ((buffer[n++] = getchar()) != '\n')
            ;
        write(clientSocket, buffer, MAX);

        bzero(buffer, MAX);
        read(clientSocket, buffer, MAX);
        printf("%s\n", buffer);
    }

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

Explanation :

Here, the client connects to the server and send the operands and operators to perform basic arithmetic operations and receives the corresponding results.

Output :


s2019103503@centos8-linux:~/
https://14.139.161.31:4200
90%
[s2019103503@centos8-linux Sat Sep 11 09:27 AM Week2]$ gcc calcServer.c -o calcServer -lpthread
[s2019103503@centos8-linux Sat Sep 11 09:27 AM Week2]$ ./calcServer
[+] Socket created.
[+] Server listening at port 3503
[+] Client 0 connected.
[+] Client 1 connected.
[+] Client 0 disconnected.
[+] Client 1 disconnected.
[s2019103503@centos8-linux Sat Sep 11 09:27 AM Week2]$ gcc calcClient.c -o calcClient -lpthread
[s2019103503@centos8-linux Sat Sep 11 09:27 AM Week2]$ ./calcClient
[+] Socket created.
[+] Connected to server.
Enter number 1 : 5
Enter number 2 : 15
Enter operation : *
Result : 75
Enter number 1 : 90
Enter number 2 : 10
Enter operation : /
Result : 9
Enter number 1 : X
[+] Disconnected from server.
[s2019103503@centos8-linux Sat Sep 11 09:28 AM Week2]$
s2019103503@centos8-linux:~/
https://14.139.161.31:4200
90%
[s2019103503@centos8-linux Sat Sep 11 09:27 AM Week2]$ ./calcClient
[+] Socket created.
[+] Connected to server.
Enter number 1 : 125
Enter number 2 : 7
Enter operation : -
Result : 118
Enter number 1 : -7
Enter number 2 : -8
Enter operation : +
Result : -15
Enter number 1 : X
[+] Disconnected from server.
[s2019103503@centos8-linux Sat Sep 11 09:28 AM Week2]$
27°C Haze
9:28 AM
11/09/2021