Motilal Nehru National Institute of TechnologyAllahabad
Prayagraj-211004 [India]

## Department of Computer Science & Engineering

**Programme Name: B.Tech** **Semester: VII** **Branch: Computer Science & Engg.**

**Course Code: CS17201** **Course Name: Distributed Systems (Lab)**

### Lab Assignment 7

Name-Sugandh Mishra
Reg-20204211
Sec- CSE C

| Lab # | Name of Experiments |
|---|---|
| 7 | **(i)** Implement RPC mechanism for a file transfer across a network in 'C'. |

Server.c----

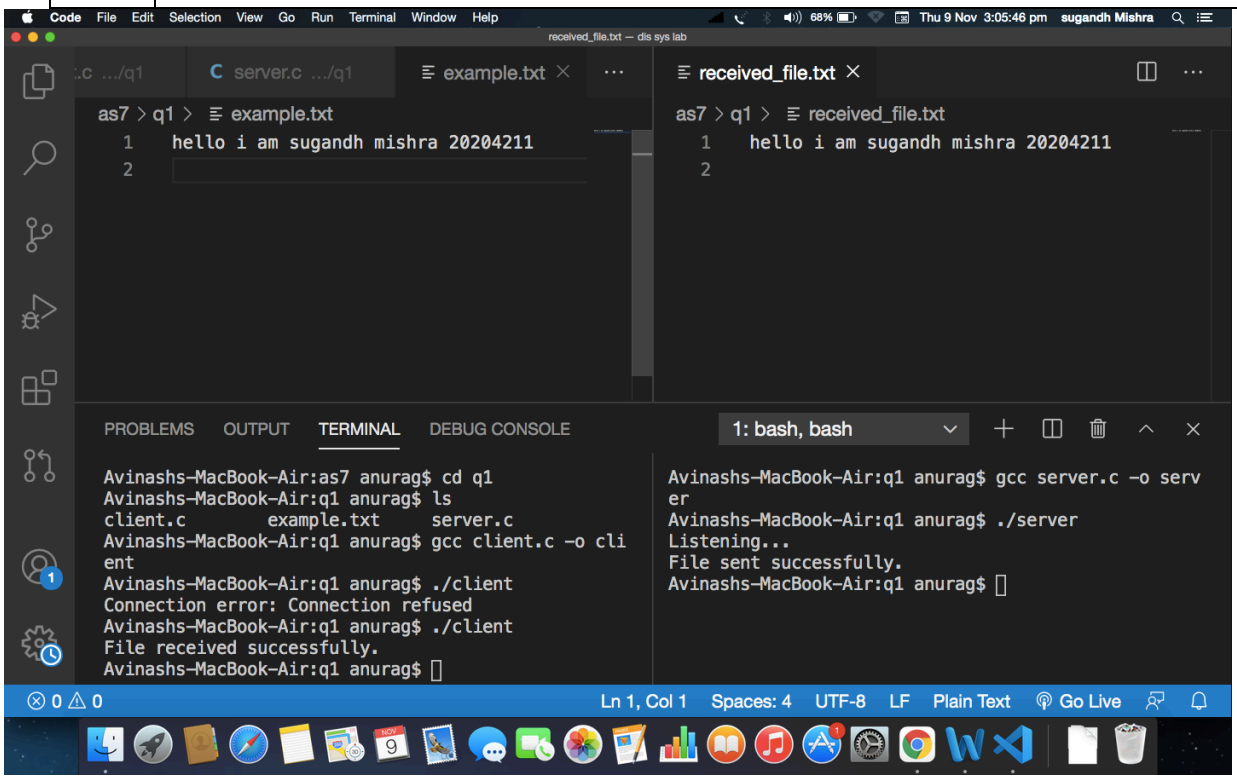
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 12345

void transfer_file(int client_socket) {
    FILE *file = fopen("example.txt", "rb");
    if (file == NULL) {
        perror("File open error");
        exit(1);
    }

    char buffer[1024];
    size_t bytesRead;

    while ((bytesRead = fread(buffer, 1, sizeof(buffer), file)) > 0) {
        send(client_socket, buffer, bytesRead, 0);
    }

    fclose(file);
}

int main() {
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_size;

    server_socket = socket(AF_INET, SOCK_STREAM, 0);
```

```
    if (server_socket < 0) {
        perror("Socket creation error");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    if (bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("Binding error");
        exit(1);
    }

    if (listen(server_socket, 10) == 0) {
        printf("Listening...\n");
    } else {
        perror("Listening error");
        exit(1);
    }

    addr_size = sizeof(client_addr);
    client_socket = accept(server_socket, (struct sockaddr*)&client_addr, &addr_size);

    transfer_file(client_socket);
    printf("File sent successfully.\n");

    close(client_socket);
    close(server_socket);
    return 0;
}
```

client.c----

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

#define SERVER_IP "127.0.0.1"  // Change this to the server's IP address
#define PORT 12345

void receive_file(int server_socket) {
    FILE *file = fopen("received_file.txt", "wb");
    if (file == NULL) {
        perror("File create error");
        exit(1);
    }

    char buffer[1024];
    int bytesRead;

    while ((bytesRead = recv(server_socket, buffer, sizeof(buffer), 0)) > 0) {
        fwrite(buffer, 1, bytesRead, file);
```

```
    }

    fclose(file);
}

int main() {
    int client_socket;
    struct sockaddr_in server_addr;

    client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket < 0) {
        perror("Socket creation error");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);

    if (connect(client_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) <
0) {
        perror("Connection error");
        exit(1);
    }

    receive_file(client_socket);
    printf("File received successfully.\n");

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



ii) Implement ‘Java RMI’ mechanism for accessing methods of remote systems.

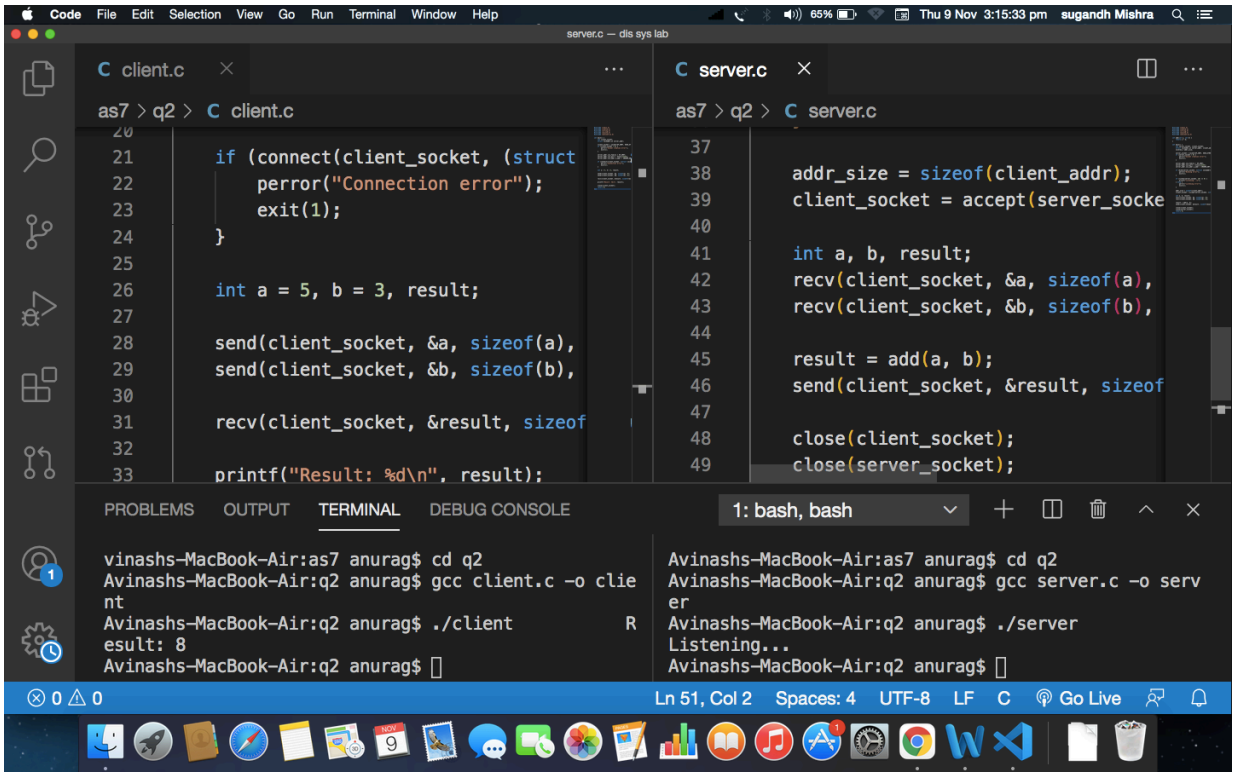
Server.c---

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>

int add(int a, int b) {
    return a + b;
}

int main() {
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_size;

    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket < 0) {
        perror("Socket creation error");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(12345);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    if (bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0)
{
        perror("Binding error");
        exit(1);
    }

    if (listen(server_socket, 10) == 0) {
        printf("Listening...\n");
    } else {
        perror("Listening error");
        exit(1);
    }

    addr_size = sizeof(client_addr);
    client_socket = accept(server_socket, (struct sockaddr*)&client_addr,
&addr_size);

    int a, b, result;
    recv(client_socket, &a, sizeof(a), 0);
    recv(client_socket, &b, sizeof(b), 0);

    result = add(a, b);
    send(client_socket, &result, sizeof(result), 0);
```

```
    close(client_socket);
    close(server_socket);
    return 0;
}
```

client.c-----

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>

int main() {
    int client_socket;
    struct sockaddr_in server_addr;

    client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket < 0) {
        perror("Socket creation error");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(12345);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    if (connect(client_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) <
0) {
        perror("Connection error");
        exit(1);
    }

    int a = 5, b = 3, result;

    send(client_socket, &a, sizeof(a), 0);
    send(client_socket, &b, sizeof(b), 0);

    recv(client_socket, &result, sizeof(result), 0);

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

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


Code File Edit Selection View Go Run Terminal Window Help
Thu 9 Nov 3:15:33 pm sugandh Mishra
server.c — dis sys lab
client.c
server.c
as7 > q2 > client.c
as7 > q2 > server.c
if (connect(client_socket, (struct
perror("Connection error");
exit(1);
}
int a = 5, b = 3, result;
send(client_socket, &a, sizeof(a),
send(client_socket, &b, sizeof(b),
recv(client_socket, &result, sizeof
printf("Result: %d\n", result);
addr_size = sizeof(client_addr);
client_socket = accept(server_socke
int a, b, result;
recv(client_socket, &a, sizeof(a),
recv(client_socket, &b, sizeof(b),
result = add(a, b);
send(client_socket, &result, sizeof
close(client_socket);
close(server_socket);
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1: bash, bash
vinashs-MacBook-Air:as7 anurag$ cd q2
Avinashs-MacBook-Air:q2 anurag$ gcc client.c -o clie
nt
Avinashs-MacBook-Air:q2 anurag$ ./client
Result: 8
Avinashs-MacBook-Air:q2 anurag$
Avinashs-MacBook-Air:as7 anurag$ cd q2
Avinashs-MacBook-Air:q2 anurag$ gcc server.c -o serv
er
Avinashs-MacBook-Air:q2 anurag$ ./server
Listening...
Avinashs-MacBook-Air:q2 anurag$
Ln 51, Col 2 Spaces: 4 UTF-8 LF C Go Live