

Name -Sumit Kumar Jha

Reg -20204212

Sec - CSE C

Motilal Nehru National Institute of Technology Allahabad Prayagraj
Distributed System (CS17201)
B.Tech (CSE) – VII Sem Lab 7

Q1. Implement RPC mechanism for a file transfer across a network in 'C'.

Code :

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;
}

```

Server.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;
}

```

The screenshot displays a VS Code editor with two open files: `example.txt` and `received_file.txt`. Both files contain the same text:

```

1 Hi,
2 Sumit Kumar Jha
3 20204212
4 DS Assignment 7
5 dated : 14th Nov 2023

```

Below the editor, the TERMINAL panel shows the execution of a C program. The left terminal (server) shows the following commands and output:

```

@sumitzha → .../Cp/DS/Assignment-7/Q1 (main) $ gcc server.c -o "server.out"
@sumitzha → .../Cp/DS/Assignment-7/Q1 (main) $ ./server.out
Listening...
File sent successfully.
@sumitzha → .../Cp/DS/Assignment-7/Q1 (main) $

```

The right terminal (client) shows the following commands and output:

```

@sumitzha → .../Cp/DS/Assignment-7/Q1 (main) $ gcc client.c -o "client.out"
@sumitzha → .../Cp/DS/Assignment-7/Q1 (main) $ ./client.out
File received successfully.
@sumitzha → .../Cp/DS/Assignment-7/Q1 (main) $

```

Q2. Implement 'Java RMI' mechanism for accessing methods of remote systems.

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("\nResult: %d\n", result);
    close(client_socket);
    return 0;
}
```

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;
}

```

The screenshot shows a Visual Studio Code editor window with the following components:

- Explorer Panel (Left):** Displays the file structure of a project named 'friendly-space-halibut-v4jx9ggjp5rhp64.github.dev'. The 'DS' folder is expanded, showing 'Q1' and 'Q2'. Under 'Q2', the files 'client.c', 'client.out', 'server.c', and 'server.out' are listed.
- Terminal Panel (Right):** Contains two terminal sessions. The first session shows the compilation of 'server.c' into 'server.out' using 'gcc server.c -o "server.out"'. The second session shows the compilation of 'client.c' into 'client.out' using 'gcc client.c -o "client.out"', followed by the execution of 'client.out' which outputs 'Result: 8'.