**Project Name:**

Online Movie Ticket Booking Using Client Server Application.

**Introduction**:

The Movie Ticket Booking System is a client-server application developed in C that enables users to view available movie shows and book tickets in real-time. This system simulates a basic ticket booking process, providing a foundational understanding of network programming concepts, including socket communication, data exchange, and client-server architecture.

**Project objective:**

The objective of this project is to implement a simple TCP-based client-server architecture where the server sends information about available movie shows to the client. The client can then make a booking request by selecting one of the shows. This project demonstrates fundamental concepts of network programming, socket communication, and basic client-server interaction.

**System Design:**

The system consists of two main components:

1. Server: The server maintains a list of available movie shows. It listens on a specific port (8080) for incoming client connections. Upon connection, the server sends the list of shows to the client and waits for a booking request. The server processes the request and sends a booking confirmation or an error message back to the client.

2. Client: The client connects to the server, receives the list of available movie shows, and sends a booking request. The client then waits for the server’s response, which either confirms the booking or indicates an error.

**Source code:**

server code (server.c)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

// Structure to represent a movie show

typedef struct {

char time[10];

char movie[20];

int price;

} MovieShow;

// Array of movie shows

MovieShow shows[] = {

{"10:00", "Avengers", rs150},

{"13:00", "Spider-Man",rs200},

{"16:00", "The Lion King",rs250},

{"19:00", "Joker", rs300}

};

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

int opt = 1;

int addrlen = sizeof(address);

// Create socket

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0) {

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Set socket options

if (setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT, &opt, sizeof(opt))) {

perror("setsockopt");

exit(EXIT\_FAILURE);

}

// Set address and port

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(PORT);

// Bind socket to address and port

if (bind(server\_fd, (struct sockaddr \*)&address, sizeof(address)) < 0) {

perror("bind failed");

exit(EXIT\_FAILURE);

}

// Listen for incoming connections

if (listen(server\_fd, 3) < 0) {

perror("listen");

exit(EXIT\_FAILURE);

}

printf("Server listening on port %d...\n", PORT);

while (1) {

// Accept incoming connection

if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t \*)&addrlen)) < 0) {

perror("accept");

exit(EXIT\_FAILURE);

}

printf("Connection accepted...\n");

// Send movie shows to client

char buffer[1024] = {0};

for (int i = 0; i < 4; i++) {

sprintf(buffer, "%s - %s (rs%d)\n", shows[i].time, shows[i].movie, shows[i].price);

send(new\_socket, buffer, strlen(buffer), 0);

}

// Receive booking request from client

read(new\_socket, buffer, 1024);

int choice = atoi(buffer);

// Process booking request

if (choice >= 1 && choice <= 4) {

sprintf(buffer, "Booking successful for %s - %s (rs%d)\n", shows[choice - 1].time, shows[choice - 1].movie, shows[choice - 1].price);

send(new\_socket, buffer, strlen(buffer), 0);

} else {

sprintf(buffer, "Invalid choice\n");

send(new\_socket, buffer, strlen(buffer), 0);

}

// Close socket

close(new\_socket);

}

    return 0;

}

**Client code:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <unistd.h>**

**#include <arpa/inet.h>**

**#define PORT 8080**

**int main() {**

**int sock = 0;**

**struct sockaddr\_in serv\_addr;**

**char buffer[1024] = {0};**

**// Create socket**

**if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {**

**printf("\n Socket creation error \n");**

**return -1;**

**}**

**// Set server address**

**serv\_addr.sin\_family = AF\_INET;**

**serv\_addr.sin\_port = htons(PORT);**

**// Convert IP address to binary format**

**if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {**

**printf("\nInvalid address/ Address not supported \n");**

**return -1;**

**}**

**// Connect to server**

**if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) {**

**printf("\nConnection Failed \n");**

**return -1;**

**}**

**printf("Connected to server...\n");**

**// Receive movie shows from server**

**printf("Available movie shows:\n");**

**for (int i = 0; i < 4; i++) {**

**read(sock, buffer, 1024);**

**printf("%s", buffer);**

**}**

**// Send booking request to server**

**printf("Enter your choice (1-4): ");**

**int choice;**

**scanf("%d", &choice);**

**sprintf(buffer, "%d", choice);**

**send(sock, buffer, strlen(buffer), 0);**

**// Receive booking confirmation from server**

**read(sock, buffer, 1024);**

**printf("%s\n", buffer);**

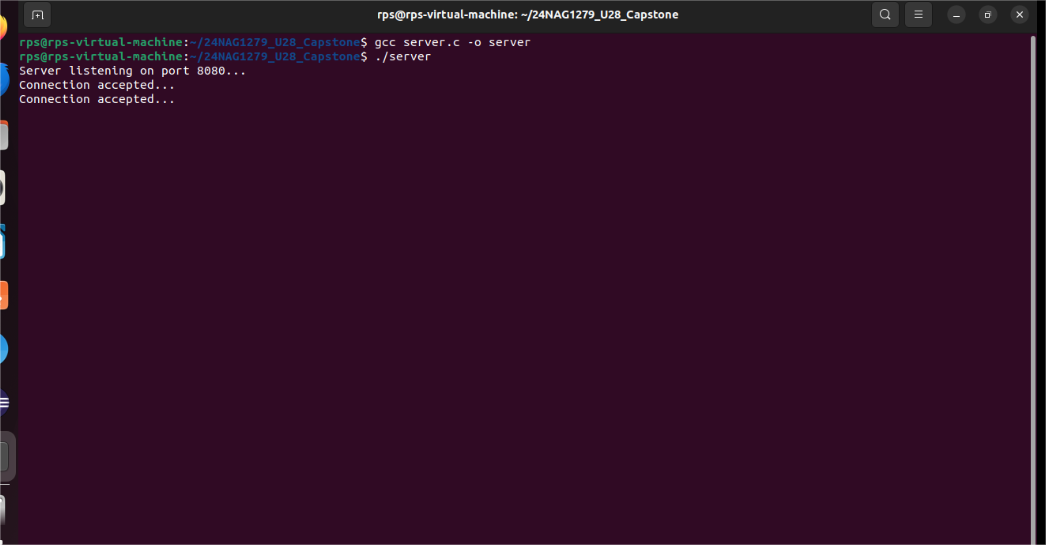
**// Close socket**

**close(sock);**

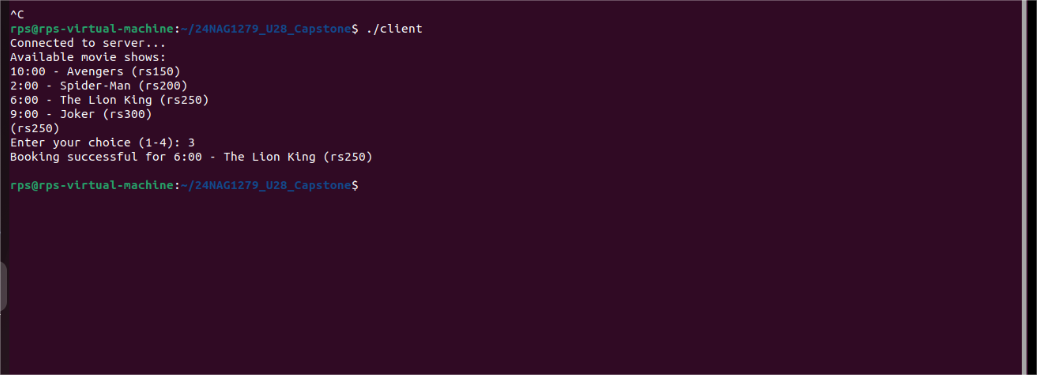
**return 0;**

**}**

**Server output:**



**Client output:**

****

**Conclusion:**

This project successfully demonstrates the implementation of a basic client-server application using C sockets. The server can handle client requests to view and book movie shows, providing a clear example of TCP communication and interaction. While basic, this project lays the foundation for more complex systems involving multiple clients, advanced booking systems, or integration with databases for persistent storage.