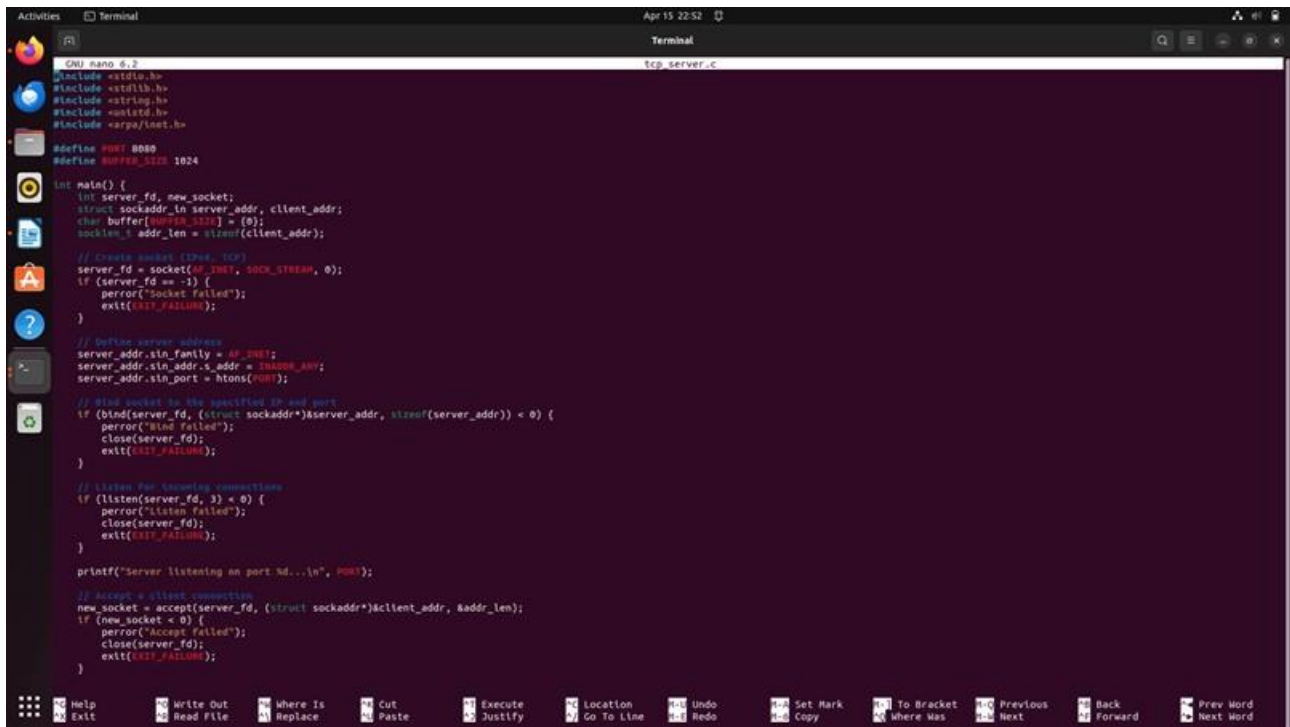


Name: Nirjala Naik

Div: B

Roll no: T512022



```
GNU nano 6.2
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int server_fd, new_socket;
    struct sockaddr_in server_addr, client_addr;
    char buffer[BUFFER_SIZE] = {0};
    socklen_t addr_len = sizeof(client_addr);

    // Create socket (Domain, Type)
    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd == -1) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

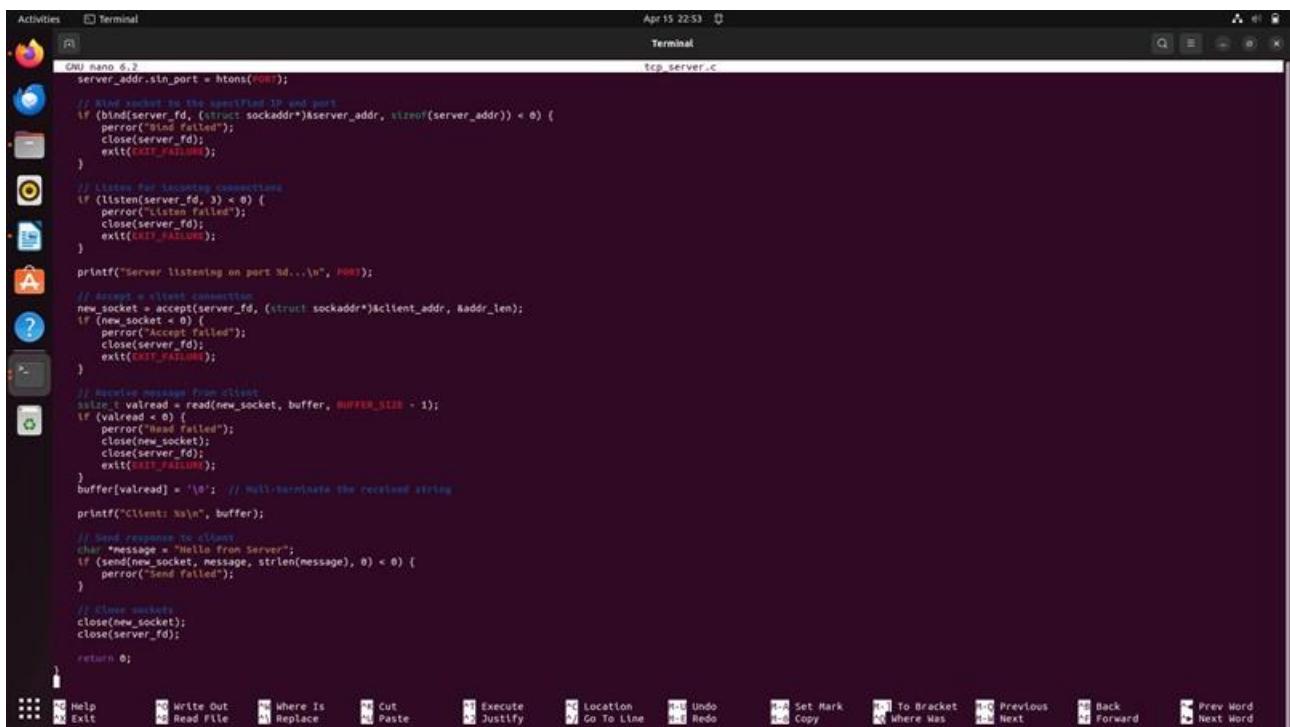
    // Define server address
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    // Bind socket to the specified IP and port
    if (bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("bind failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(server_fd, 3) < 0) {
        perror("listen failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

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

    // Accept a client connection
    new_socket = accept(server_fd, (struct sockaddr*)&client_addr, &addr_len);
    if (new_socket < 0) {
        perror("accept failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }
}
```



```
server_addr.sin_port = htons(PORT);

// Bind socket to the specified IP and port
if (bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
    perror("bind failed");
    close(server_fd);
    exit(EXIT_FAILURE);
}

// Listen for incoming connections
if (listen(server_fd, 3) < 0) {
    perror("listen failed");
    close(server_fd);
    exit(EXIT_FAILURE);
}

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

// Accept a client connection
new_socket = accept(server_fd, (struct sockaddr*)&client_addr, &addr_len);
if (new_socket < 0) {
    perror("accept failed");
    close(server_fd);
    exit(EXIT_FAILURE);
}

// Receive message from client
ssize_t valread = read(new_socket, buffer, BUFFER_SIZE - 1);
if (valread < 0) {
    perror("read failed");
    close(new_socket);
    close(server_fd);
    exit(EXIT_FAILURE);
}
buffer[valread] = '\0'; // Null-terminate the received string
printf("Client: %s\n", buffer);

// Send response to client
char *message = "Hello from Server";
if (send(new_socket, message, strlen(message), 0) < 0) {
    perror("send failed");
}

// Close sockets
close(new_socket);
close(server_fd);

return 0;
}
```

```
Activities Terminal Apr 15 22:54 tcp_client.c
Ctrl nano 0.2
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024
int main() {
    int sock;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE] = {0};
    // Create socket (IPv4, TCP)
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }
    // Define server address
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    // Connect to server
    if (connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("Connection failed");
        close(sock);
        exit(EXIT_FAILURE);
    }
    // Send message to server
    char *message = "Hello from client";
    if (send(sock, message, strlen(message), 0) < 0) {
        perror("Send failed");
        close(sock);
        exit(EXIT_FAILURE);
    }
    // Receive response from server
    ssize_t valread = read(sock, buffer, BUFFER_SIZE - 1);
    if (valread < 0) {
        perror("Read failed");
        close(sock);
        exit(EXIT_FAILURE);
    }
    buffer[valread] = '\0'; // Null-terminate the received string
    printf("Server: %s\n", buffer);
    // Close socket
    close(sock);
    return 0;
}
```

```
Activities Terminal Apr 15 22:49
\Student@pc:~$ cd T511069
\Student@pc:~$ nano tcp_server.c
\Student@pc:~$ nano tcp_client.c
\Student@pc:~$ gcc tcp_server.c -o tcp_server
\Student@pc:~$ gcc tcp_client.c -o tcp_client
\Student@pc:~$ ./tcp_server
Server listening on port 8080...
Client: Hello from client
\Student@pc:~$
```

