

## CODE FOR CILENT

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <unistd.h>
```

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

```
#include <sys/socket.h>
```

```
#include <pthread.h>
```

```
#include <time.h>
```

```
#define PORT 5555
```

```
#define BUFFER_SIZE 1024
```

```
#define TIMEOUT_SECONDS 10
```

```
char shared_resource[BUFFER_SIZE];
```

```
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
```

```
void *handle_client(void *arg) {
```

```
    int client_socket = *((int *)arg);
```

```
    char buffer[BUFFER_SIZE];
```

```
    time_t start_time, current_time;
```

```
    // Get the start time
```

```
    time(&start_time);
```

```

while (1) {
    // Check if the time limit has been reached
time(&current_time);
    if (difftime(current_time, start_time) > TIMEOUT_SECONDS) {
        printf("Timeout reached. Closing connection.\n");
        break;
    }

    int bytes_received = recv(client_socket, buffer, BUFFER_SIZE, 0);
    if (bytes_received <= 0) {
        break;
    }
    buffer[bytes_received] = '\0';

    pthread_mutex_lock(&lock);
    strcpy(shared_resource, buffer);
    printf("Received message from client: %s\n", buffer);
    printf("Shared resource: %s\n", shared_resource);
    pthread_mutex_unlock(&lock);

    send(client_socket, buffer, strlen(buffer), 0);
}

```

```
    close(client_socket);  
    pthread_exit(NULL);  
}
```

```
int main() {  
    int server_socket, client_socket;  
int server_socket, client_socket;  
    struct sockaddr_in server_addr, client_addr;  
    socklen_t client_addr_len = sizeof(client_addr);  
  
    // Create socket  
    server_socket = socket(AF_INET, SOCK_STREAM, 0);  
    if (server_socket == -1) {  
        perror("Socket creation failed");  
        exit(EXIT_FAILURE);  
    }  
  
    // Bind socket  
    server_addr.sin_family = AF_INET;  
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);  
    server_addr.sin_port = htons(PORT);  
    if (bind(server_socket, (struct sockaddr *)&server_addr,  
sizeof(server_addr)) == -1) {  
        perror("Binding failed");  
    }  
}
```

```

        exit(EXIT_FAILURE);
    }

    // Listen for connections
    if (listen(server_socket, 5) == -1) {
        perror("Listening failed");
        exit(EXIT_FAILURE);
    }

    printf("Server listening on port %d\n", PORT);
    while (1) {
        // Accept client connection
        client_socket = accept(server_socket, (struct sockaddr
*)&client_addr, &client_addr_len);

        if (client_socket == -1) {
            perror("Acceptance failed");
            exit(EXIT_FAILURE);
        }

        printf("Connection accepted from %s:%d\n",
inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));

        // Create a new thread to handle client
        pthread_t client_thread;

        if (pthread_create(&client_thread, NULL, handle_client,
&client_socket) != 0) {
            perror("Thread creation failed");

```

```
        exit(EXIT_FAILURE);
    }
}

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

## CODE FOR SERVER

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/select.h>

#define PORT 5555
#define BUFFER_SIZE 1024
#define TIMEOUT_SEC 5

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

```
char buffer[BUFFER_SIZE];

// Create socket
client_socket = socket(AF_INET, SOCK_STREAM, 0);
if (client_socket == -1) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

// Connect to server
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
server_addr.sin_port = htons(PORT);
if (connect(client_socket, (struct sockaddr *)&server_addr,
sizeof(server_addr)) == -1) {
    perror("Connection failed");
    exit(EXIT_FAILURE);
}

fd_set readfds;
struct timeval timeout;
int select_result;

while (1) {
```

```
// Clear the socket set
FD_ZERO(&readfds);

// Add client socket to the set
FD_SET(client_socket, &readfds);


// Set timeout
timeout.tv_sec = TIMEOUT_SEC;
timeout.tv_usec = 0;


// Wait for socket activity with timeout
select_result = select(client_socket + 1, &readfds, NULL, NULL,
&timeout);
if (select_result == -1) {
    perror("Select error");
    break;
} else if (select_result == 0) {
    printf("Timeout occurred. No response from server.\n");
    break;
} else {
    if (FD_ISSET(client_socket, &readfds)) {
        // Send message to server
        printf("Enter message to send to server: ");
        fgets(buffer, BUFFER_SIZE, stdin);
        if (strcmp(buffer, "exit\n") == 0) {
```

```
        break;
    }
    send(client_socket, buffer, strlen(buffer), 0);

    // Receive response from server
    int bytes_received = recv(client_socket, buffer,
BUFFER_SIZE, 0);
    if (bytes_received <= 0) {
        printf("Connection closed by server.\n");
        break;
    }
    buffer[bytes_received] = '\0';
    printf("Received response from server: %s\n", buffer);
}
}
}

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

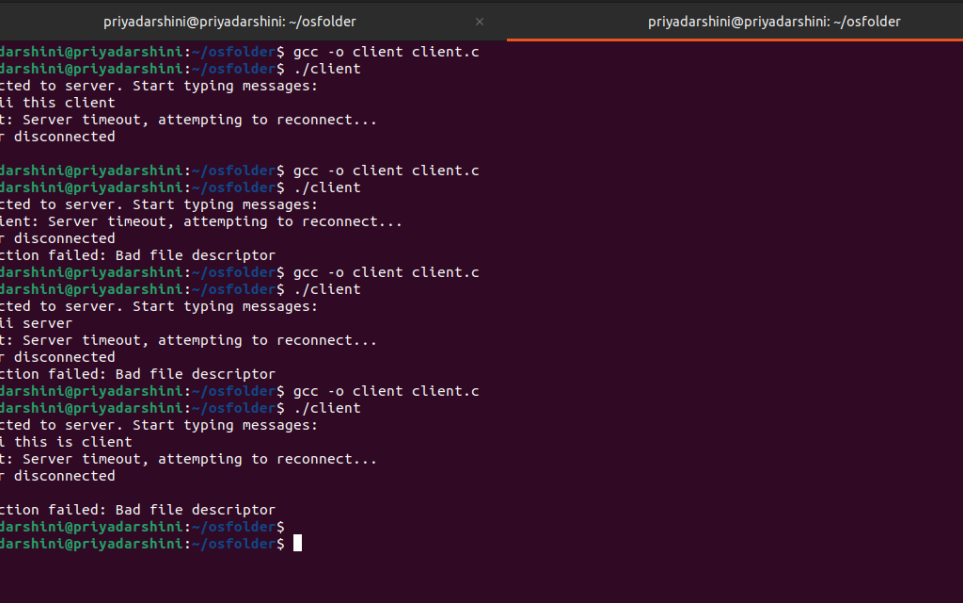


## OUTPUT FOR CILENT

The screenshot shows a Linux desktop with a dark purple background. On the left is a vertical sidebar with icons for Activities, Terminal, Firefox, Telegram, Files, GIMP, LibreOffice Writer, App Store, and a help icon. The top panel displays 'Activities', 'Terminal', and the date/time 'Mar 25 12:31'. A terminal window is open, showing the following commands and output:

```
priyadarshini@priyadarshini: ~/osfolder
priyadarshini@priyadarshini:~/osfolder$ cd osfolder
priyadarshini@priyadarshini:~/osfolder$ open server.c
priyadarshini@priyadarshini:~/osfolder$ open client.c
priyadarshini@priyadarshini:~/osfolder$ gcc -o server server.c -lpthread
priyadarshini@priyadarshini:~/osfolder$ ./server
Server started, listening on port 9999
Accepted connection from client
Received from client: hiii this client
Client disconnected
Accepted connection from client
Client disconnected
Accepted connection from client
Received from client: hiii server
Client disconnected
Accepted connection from client
Received from client: hii this is client
Client disconnected
```

## OUTPUT FOR SERVER



The screenshot shows a macOS desktop with a dark theme. At the top, there is a status bar with the date and time 'Mar 25 12:32' and system icons for network, volume, and battery. Below the status bar, there is a dock with several application icons: Finder, Launchpad, Safari, Mail, Messages, Photos, and a custom icon. The main window is a Terminal application titled 'Terminal'. The terminal window has a dark background and a light-colored text area. The text in the terminal shows the user running a C program. The program is a simple client-server chat application. The user has compiled the program and is running it. The output shows the client connecting to the server, sending messages, and the server responding. The user has run the program multiple times, and each time it successfully connects and communicates.

```
Mar 25 12:32
priyadarshini@priyadarshini: ~/osfolder
priyadarshini@priyadarshini:~/osfolder$ gcc -o client client.c
priyadarshini@priyadarshini:~/osfolder$ ./client
Connected to server. Start typing messages:
>> hiii this client
Client: Server timeout, attempting to reconnect...
Server disconnected
^C
priyadarshini@priyadarshini:~/osfolder$ gcc -o client client.c
priyadarshini@priyadarshini:~/osfolder$ ./client
Connected to server. Start typing messages:
>> Client: Server timeout, attempting to reconnect...
Server disconnected
Connection failed: Bad file descriptor
priyadarshini@priyadarshini:~/osfolder$ gcc -o client client.c
priyadarshini@priyadarshini:~/osfolder$ ./client
Connected to server. Start typing messages:
>> hiii server
Client: Server timeout, attempting to reconnect...
Server disconnected
Connection failed: Bad file descriptor
priyadarshini@priyadarshini:~/osfolder$ gcc -o client client.c
priyadarshini@priyadarshini:~/osfolder$ ./client
Connected to server. Start typing messages:
>> hii this is client
Client: Server timeout, attempting to reconnect...
Server disconnected
Connection failed: Bad file descriptor
priyadarshini@priyadarshini:~/osfolder$
priyadarshini@priyadarshini:~/osfolder$
```

