Simple Chat Application

To be proficient in developing chat application between two users using Client Server Connection.

server.c

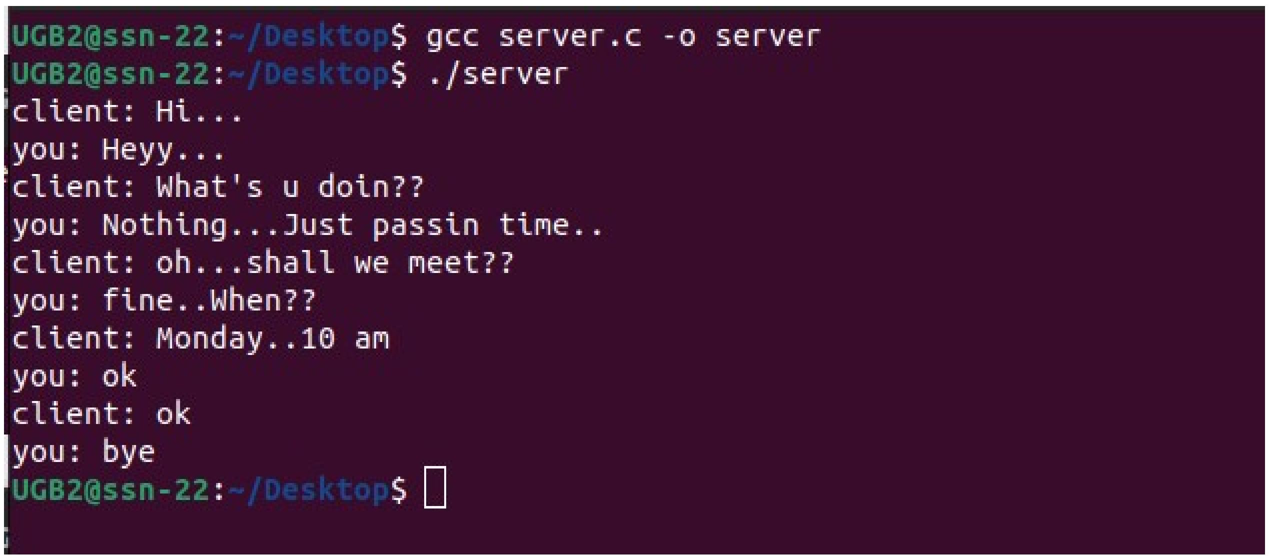
```
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#define PORT 8080
int main(int argc, char const* argv[])
{
      int sd,newfd;
      struct sockaddr_in server;
      int serverlen= sizeof(server);
      char buffer[1024]={0};
      sd=socket(AF_INET,SOCK_STREAM,0);
      server.sin_family=AF_INET;
      server.sin_port=htons(PORT);
      server.sin_addr.s_addr=INADDR_ANY;
      if(bind(sd,(struct
sockaddr*)&server,sizeof(server))<0){
        perror("bind");
        exit(1);
      }
      if(listen(sd,5)<0){
        perror("Listen");
        exit(1);
      }
      newfd=accept(sd,(struct sockaddr*)&server,
(socklen_t*)&serverlen);
      while(1){
        memset(buffer,0,1024);
        read(newfd,buffer,1024);
        if(strcmp(buffer,"bye")==0){
            break;
        }
        printf("client: %s\n",buffer);
        printf("you: ");
        fgets(buffer,1024,stdin);
        buffer[strcspn(buffer,"\n")]='\0';
```

```
           write(newfd,buffer,strlen(buffer));
        }
        close(newfd);
        close(sd);
}
```

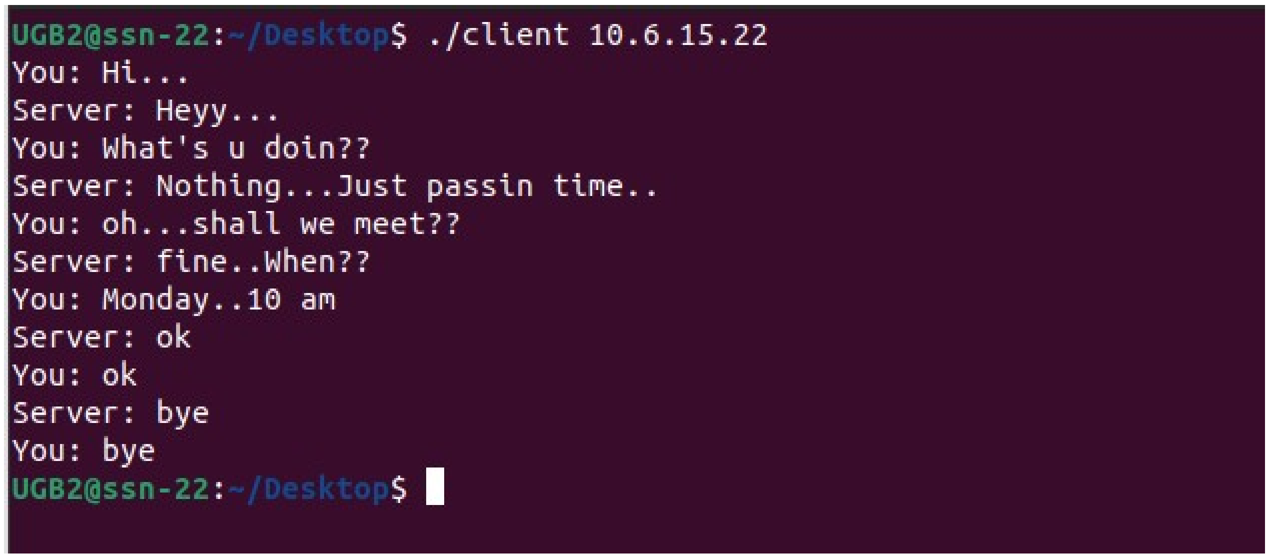
client.c

```
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#define PORT 8080

int main(int argc, char const* argv[])
{
     int sock;
     struct sockaddr_in serv_addr;
     char buffer[1024]={0};
     sock= socket(AF_INET,SOCK_STREAM,0);
     serv_addr.sin_family= AF_INET;
     serv_addr.sin_port=htons(PORT);
     serv_addr.sin_addr.s_addr=inet_addr(argv[1]);

     connect(sock,(struct
sockaddr* )&serv_addr,sizeof(serv_addr));
     while(1){
        printf("You: ");
        fgets(buffer,1024,stdin);
        buffer[strcspn(buffer,"\n")]=0;
        write(sock,buffer,strlen(buffer));
        if(strcmp(buffer,"bye")==0) break;
        memset(buffer,0,1024);
        read(sock,buffer,1024);
        printf("Server: %s\n",buffer);

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

Output:



```
UGB2@ssn-22:~/Desktop$ gcc server.c -o server
UGB2@ssn-22:~/Desktop$ ./server
client: Hi...
you: Heyy...
client: What's u doin??
you: Nothing...Just passin time..
client: oh...shall we meet??
you: fine..When??
client: Monday..10 am
you: ok
client: ok
you: bye
UGB2@ssn-22:~/Desktop$
```



```
UGB2@ssn-22:~/Desktop$ ./client 10.6.15.22
You: Hi...
Server: Heyy...
You: What's u doin??
Server: Nothing...Just passin time..
You: oh...shall we meet??
Server: fine..When??
You: Monday..10 am
Server: ok
You: ok
Server: bye
You: bye
UGB2@ssn-22:~/Desktop$
```

Multiuser Chat Application

To be proficient in developing chat application between four users using Client Server Connection.

server.c

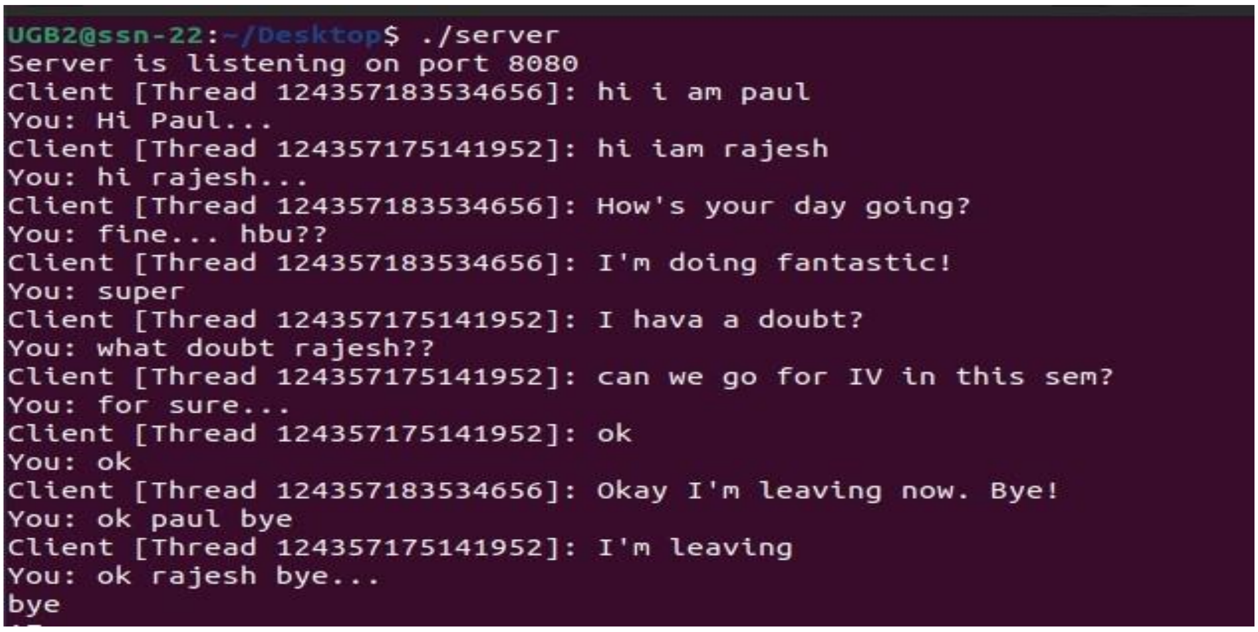
```
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <pthread.h>

#define PORT 8080
#define BUFFER_SIZE 1024

typedef struct {
    int client_socket;
    pthread_t thread_id;
} client_info_t;

void* handle_client(void* arg) {
    client_info_t* info = (client_info_t*)arg;
    int client_socket = info->client_socket;
    char buffer[BUFFER_SIZE];
    memset(buffer, 0, BUFFER_SIZE);

    while (1) {
        int bytes_read = read(client_socket, buffer,
BUFFER_SIZE - 1);
        if (bytes_read <= 0) {
            // Connection closed or error
            break;
        }
        buffer[bytes_read] = '\0';  // Null-terminate the
buffer
        if (strcmp(buffer, "bye") == 0) {
            break;
        }
        printf("Client [Thread %lu]: %s\n", info-
>thread_id, buffer);

        // Echo the message back to the client
        printf("You: ");
        memset(buffer, 0, BUFFER_SIZE);
        fgets(buffer, BUFFER_SIZE, stdin);
        buffer[strcspn(buffer, "\n")] = '\0'; // Remove
newline character
        write(client_socket, buffer, strlen(buffer));
    }

    close(client_socket);
    free(info); // Free allocated memory for
client_info_t
    return NULL;
}

int main(int argc, char const* argv[]) {
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_addr_len = sizeof(client_addr);

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

    if (listen(server_socket, 5) < 0) {
        perror("Listen failed");
        exit(1);
    }

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

    while (1) {
        client_socket = accept(server_socket, (struct
sockaddr*)&client_addr, &client_addr_len);
        if (client_socket < 0) {
            perror("Accept failed");
            continue;
        }

        client_info_t* info =
malloc(sizeof(client_info_t));
        if (info == NULL) {
            perror("Memory allocation failed");
            close(client_socket);
            continue;
        }
        info->client_socket = client_socket;

        // Create a thread to handle the client
        if (pthread_create(&info->thread_id, NULL,
handle_client, (void*)info) != 0) {
            perror("Failed to create thread");
            free(info);
            close(client_socket);
            continue;
        }

        pthread_detach(info->thread_id); // Detach the
thread to clean up resources automatically
    }

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

client.c

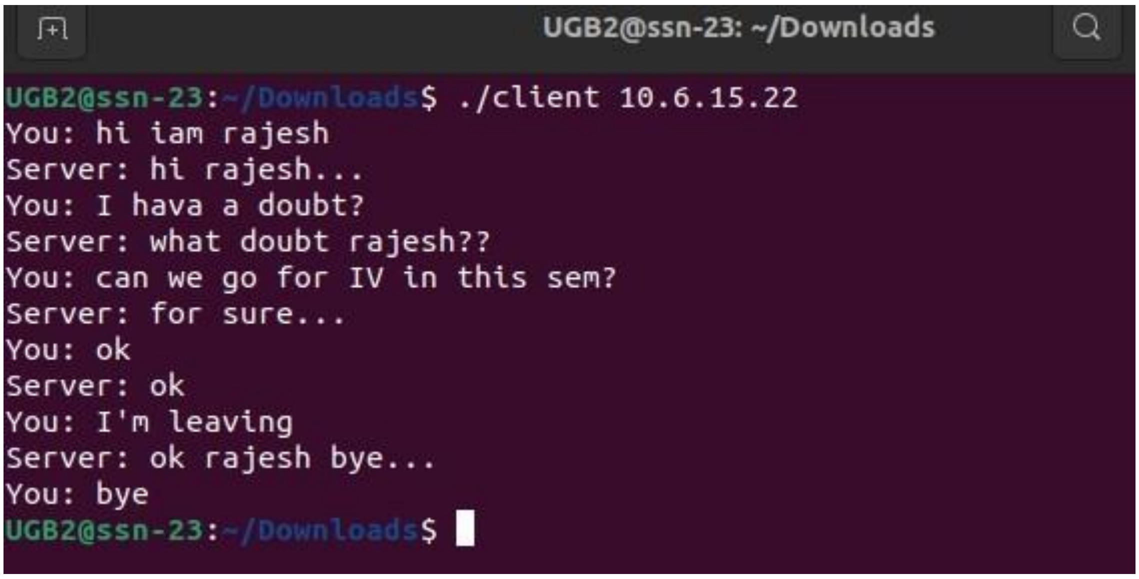
```
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#define PORT 8080

int main(int argc, char const* argv[])
{
     int sock;
```

```
      struct sockaddr_in serv_addr;
      char buffer[1024]={0};
      sock= socket(AF_INET,SOCK_STREAM,0);
      serv_addr.sin_family= AF_INET;
      serv_addr.sin_port=htons(PORT);
      serv_addr.sin_addr.s_addr=inet_addr(argv[1]);

      connect(sock,(struct
sockaddr* )&serv_addr,sizeof(serv_addr));
      while(1){
         printf("You: ");
         fgets(buffer,1024,stdin);
         buffer[strcspn(buffer,"\n")]=0;
         write(sock,buffer,strlen(buffer));
         if(strcmp(buffer,"bye")==0) break;
         memset(buffer,0,1024);
         read(sock,buffer,1024);
         printf("Server: %s\n",buffer);

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

Output:

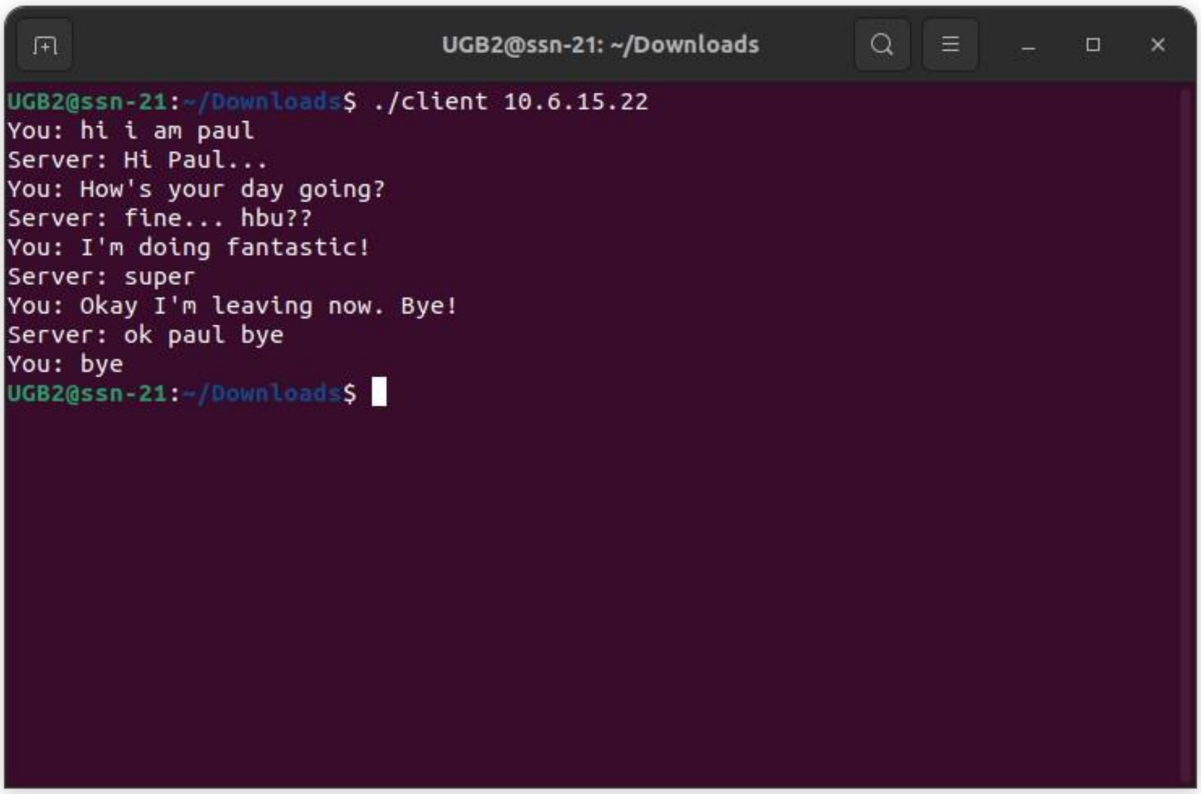


```
UGB2@ssn-22:~/Desktop$ ./server
Server is listening on port 8080
Client [Thread 124357183534656]: hi i am paul
You: Hi Paul...
Client [Thread 124357175141952]: hi iam rajesh
You: hi rajesh...
Client [Thread 124357183534656]: How's your day going?
You: fine... hbu??
Client [Thread 124357183534656]: I'm doing fantastic!
You: super
Client [Thread 124357175141952]: I hava a doubt?
You: what doubt rajesh??
Client [Thread 124357175141952]: can we go for IV in this sem?
You: for sure...
Client [Thread 124357175141952]: ok
You: ok
Client [Thread 124357183534656]: Okay I'm leaving now. Bye!
You: ok paul bye
Client [Thread 124357175141952]: I'm leaving
You: ok rajesh bye...
bye
```



UGB2@ssn-23: ~/Downloads

```
UGB2@ssn-23:~/Downloads$ ./client 10.6.15.22
You: hi iam rajesh
Server: hi rajesh...
You: I hava a doubt?
Server: what doubt rajesh??
You: can we go for IV in this sem?
Server: for sure...
You: ok
Server: ok
You: I'm leaving
Server: ok rajesh bye...
You: bye
UGB2@ssn-23:~/Downloads$
```



UGB2@ssn-21: ~/Downloads

```
UGB2@ssn-21:~/Downloads$ ./client 10.6.15.22
You: hi i am paul
Server: Hi Paul...
You: How's your day going?
Server: fine... hbu??
You: I'm doing fantastic!
Server: super
You: Okay I'm leaving now. Bye!
Server: ok paul bye
You: bye
UGB2@ssn-21:~/Downloads$
```