The client code

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

int main(int argc, char const* argv[])
{
 const int port = 3400;
 const char *host_service = "127.0.0.1";
 int socket_sock_cn = 0, value_read, client_instance;
 struct sockaddr_in serv_addr;
 char* hello = "This is the message from the client";
 char buffer[1024] = { 0 };
 if ((socket_sock_cn = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
   printf("\n Could not create the socket connection\n");
   return -1;
 }

 serv_addr.sin_family = AF_INET;
 serv_addr.sin_port = htons(port);

 // Convert IPv4 and IPv6 addresses from text to binary
 if (inet_pton(AF_INET, host_service, &serv_addr.sin_addr)
   <= 0) {
   printf(
     "\n The address format passed is not supported \n");
   return -1;
 }

 if ((client_instance
   = connect(socket_sock_cn, (struct sockaddr*)&serv_addr,
       sizeof(serv_addr)))
   < 0) {
   printf("\nConnection Failed \n");
   return -1;
 }
 send(socket_sock_cn , hello, strlen(hello), 0);
 printf("Message from the client has been sent to the server\n");
```

```
 value_read = read(socket_sock_cn, buffer, 1024);
 printf("%s\n", buffer);

 // closing the connected socket
 close(client_instance);
  // return an integer value as the function is of type integer
 return 5;
}
```

The server code

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


int main(int argc, char const *argv[])
{
 const int port = 3400;
 int server_intance, new_socket, value_read;
 struct sockaddr_in address;
 int opt = 1;
 int addrlen = sizeof(address);
 char buffer[1024] = {0};
 char *hello = "Hello from the server :)";

 // Creating socket file
 if ((server_intance = socket(AF_INET, SOCK_STREAM, 0)) < 0)
 {
   perror("socket failed");
   exit(EXIT_FAILURE);
 }

 // Forcefully attaching socket to the port 3400
 if (setsockopt(server_intance, SOL_SOCKET,
                SO_REUSEADDR | SO_REUSEPORT, &opt,
                sizeof(opt)))
```

```
{
  perror("setsockopt");
  exit(EXIT_FAILURE);
}
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons(port);

// Forcefully attaching socket to the port 3400
if (bind(server_intance, (struct sockaddr *)&address,
         sizeof(address)) < 0)
{
  perror("bind failed");
  exit(EXIT_FAILURE);
}
if (listen(server_intance, 3) < 0)
{
  perror("listen");
  exit(EXIT_FAILURE);
}
if ((new_socket = accept(server_intance, (struct sockaddr *)&address,
                         (socklen_t *)&addrlen)) < 0)
{
  perror("accept");
  exit(EXIT_FAILURE);
}
value_read = read(new_socket, buffer, 1024);
printf("%s\n", buffer);
send(new_socket, hello, strlen(hello), 0);
printf("Hello message sent\n");

// closing the connected socket
close(new_socket);
// closing the listening socket
shutdown(server_intance, SHUT_RDWR);

// return an integer value as the function is of type integer
return 5;
}
```

Code Results
Starting the server

```
ezra@ezrakeeps ~/coding/classCode/3.1/assignments/system_prog/sockets
 % gcc server.c -o ./server && ./server
```

Send data from the client

```
ezra@ezrakeeps ~/coding/classCode/3.1/assignments/system_prog/sockets
 % gcc client.c -o ./client && ./client
Message from the client has been sent to the server
Hello from the server :)
ezra@ezrakeeps ~/coding/classCode/3.1/assignments/system_prog/sockets
 %
```

Response from server

```
ezra@ezrakeeps ~/coding/classCode/3.1/assignments/system_prog/sockets
 % gcc server.c -o ./server && ./server
This is the message from the client
Hello message sent
ezra@ezrakeeps ~/coding/classCode/3.1/assignments/system_prog/sockets
 %
```