

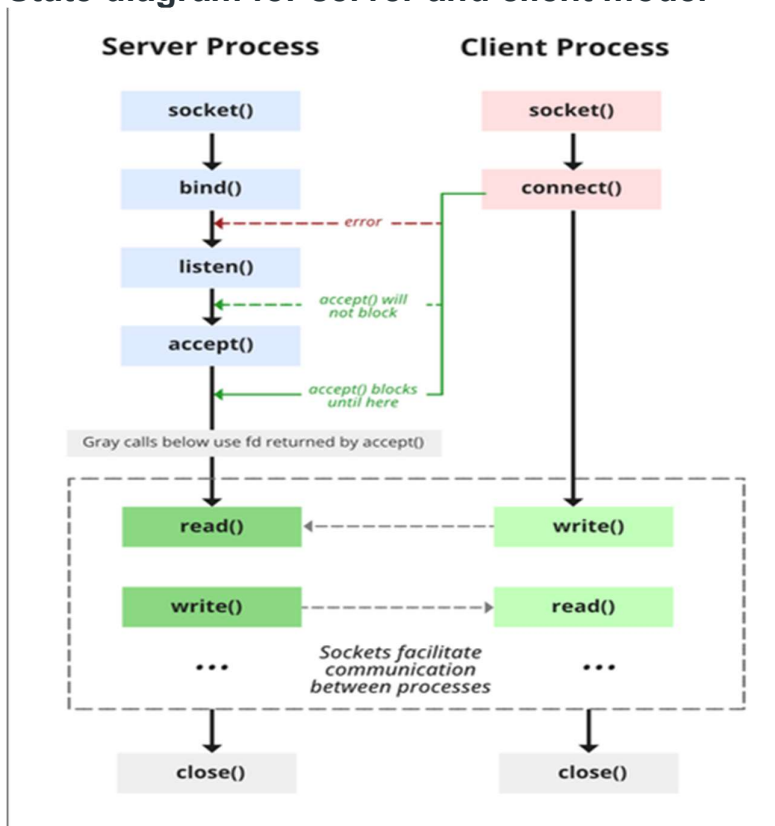
NAME – ATUL SHARMA  
UID – 2021700060  
BATCH – D4  
Date :- 3/3/23

**AIM :-** Develop client-server model using Socket Programming for given scenario.

## What is socket programming?

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

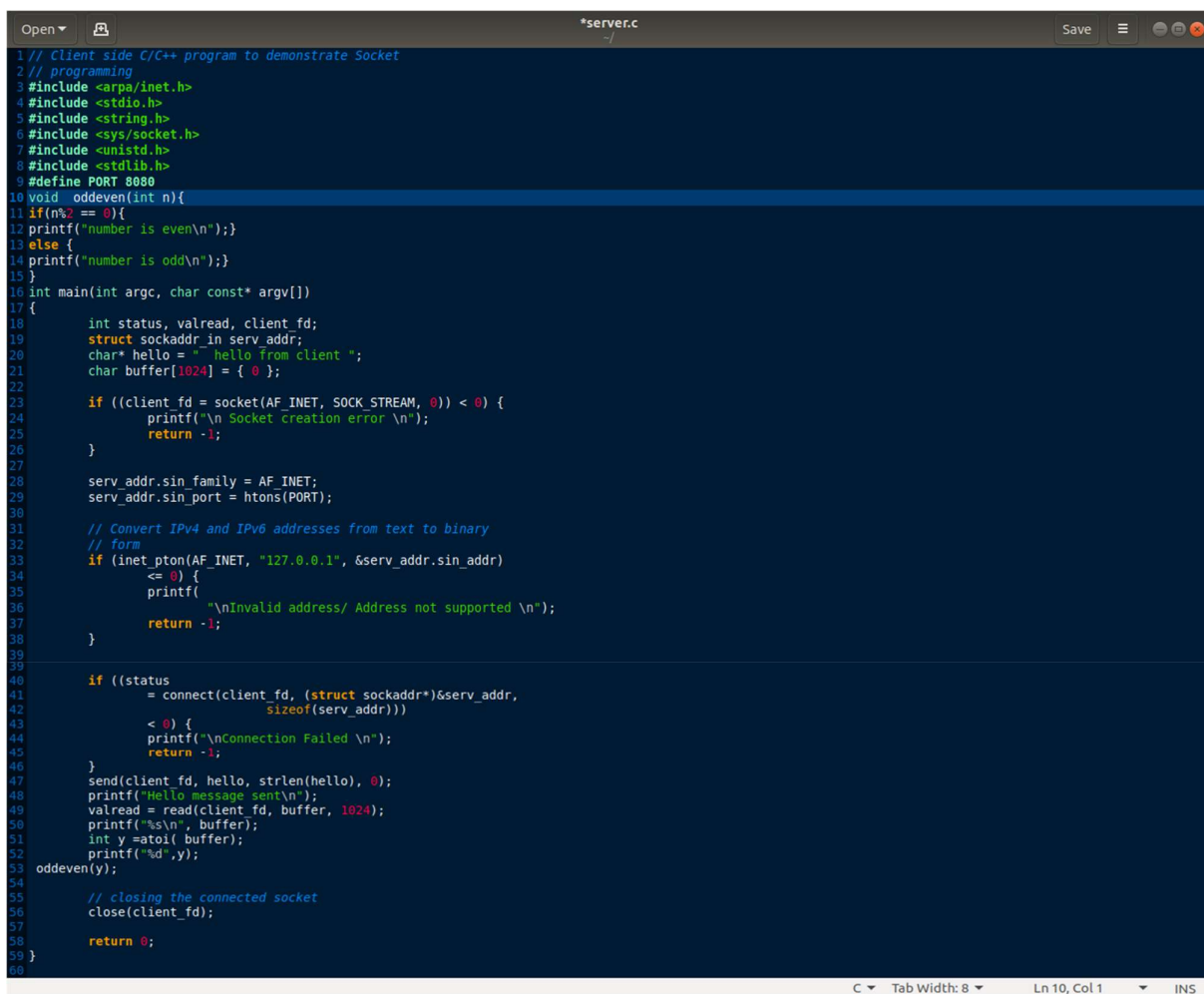
## State diagram for server and client model



## Implementation

Here we are exchanging one hello message between server and client to demonstrate the client/server model. And check no. whether a number is odd or even by making function in server and got output in client.

### server.c



```
1 // Client side C/C++ program to demonstrate Socket
2 // programming
3 #include <arpa/inet.h>
4 #include <stdio.h>
5 #include <string.h>
6 #include <sys/socket.h>
7 #include <unistd.h>
8 #include <stdlib.h>
9 #define PORT 8080
10 void oddeven(int n){
11     if(n%2 == 0){
12         printf("number is even\n");
13     }
14     else {
15         printf("number is odd\n");
16     }
17 }
18 int main(int argc, char const* argv[])
19 {
20     int status, valread, client_fd;
21     struct sockaddr_in serv_addr;
22     char* hello = "Hello from client ";
23     char buffer[1024] = { 0 };
24
25     if ((client_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
26         printf("\n Socket creation error \n");
27         return -1;
28     }
29
30     serv_addr.sin_family = AF_INET;
31     serv_addr.sin_port = htons(PORT);
32
33     // Convert IPv4 and IPv6 addresses from text to binary form
34     if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)
35         <= 0) {
36         printf(
37             "\nInvalid address/ Address not supported \n");
38         return -1;
39     }
40
41     if ((status
42         = connect(client_fd, (struct sockaddr*)&serv_addr,
43             sizeof(serv_addr)))
44         < 0) {
45         printf("\nConnection Failed \n");
46         return -1;
47     }
48     send(client_fd, hello, strlen(hello), 0);
49     printf("Hello message sent\n");
50     valread = read(client_fd, buffer, 1024);
51     printf("%s\n", buffer);
52     int y = atoi(buffer);
53     printf("%d", y);
54     oddeven(y);
55
56     // closing the connected socket
57     close(client_fd);
58
59     return 0;
60 }
```

```
// Client side C/C++ program to demonstrate Socket
// programming
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
```

```

#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>
#define PORT 8080
void oddeven(int n){
if(n%2 == 0){
printf("number is even\n");}
else {
printf("number is odd\n");}
}
int main(int argc, char const* argv[])
{
    int status, valread, client_fd;
    struct sockaddr_in serv_addr;
    char* hello = " hello from client ";
    char buffer[1024] = { 0 };

    if ((client_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\n Socket creation error \n");
        return -1;
    }

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

    // Convert IPv4 and IPv6 addresses from text to binary
    // form
    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)
        <= 0) {
        printf(
            "\nInvalid address/ Address not supported \n");
        return -1;
    }

    if ((status
        = connect(client_fd, (struct sockaddr*)&serv_addr,
            sizeof(serv_addr)))
        < 0) {
        printf("\nConnection Failed \n");
        return -1;
    }
    send(client_fd, hello, strlen(hello), 0);
    printf("Hello message sent\n");
    valread = read(client_fd, buffer, 1024);
    printf("%s\n", buffer);
    int y = atoi( buffer);
    printf("%d",y);
    oddeven(y);

    // closing the connected socket
    close(client_fd);
}

```

```

    return 0;
}

```

## client.c – file

```

1 // Server side C/C++ program to demonstrate Socket
2 // programming
3 #include <netinet/in.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7 #include <sys/socket.h>
8 #include <unistd.h>
9 #define PORT 8080
10 int main(int argc, char const* argv[])
11 {
12     int server_fd, new_socket, valread;
13     struct sockaddr_in address;
14     int opt = 1;
15     int addrlen = sizeof(address);
16     char buffer[1024] = { 0 };
17     char* hello = "3";
18
19     // Creating socket file descriptor
20     if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
21         perror("socket failed");
22         exit(EXIT_FAILURE);
23     }
24
25     // Forcefully attaching socket to the port 8080
26     if (setsockopt(server_fd, SOL_SOCKET,
27         SO_REUSEADDR | SO_REUSEPORT, &opt,
28         sizeof(opt))) {
29         perror("setsockopt");
30         exit(EXIT_FAILURE);
31     }
32     address.sin_family = AF_INET;
33     address.sin_addr.s_addr = INADDR_ANY;
34     address.sin_port = htons(PORT);
35
36     // Forcefully attaching socket to the port 8080
37     if (bind(server_fd, (struct sockaddr*)&address,
38         sizeof(address))
39         < 0) {
40         perror("bind failed");
41         exit(EXIT_FAILURE);
42     }
43     if (listen(server_fd, 3) < 0) {
44         perror("listen");
45         exit(EXIT_FAILURE);
46     }
47     if ((new_socket
48         = accept(server_fd, (struct sockaddr*)&address,
49         (socklen_t*)&addrlen))
50         < 0) {
51         perror("accept");
52         exit(EXIT_FAILURE);
53     }
54     valread = read(new_socket, buffer, 1024);
55     printf("%s\n", buffer);
56     send(new_socket, hello, strlen(hello), 0);
57     printf("Hello message sent\n");
58
59     // closing the connected socket
60     close(new_socket);
61     // closing the listening socket
62     shutdown(server_fd, SHUT_RDWR);
63     return 0;
64 }
65

```

```

/ Server side C/C++ program to demonstrate Socket
// programming
#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 server_fd, new_socket, valread;

```

```

struct sockaddr_in address;
int opt = 1;
int addrlen = sizeof(address);
char buffer[1024] = { 0 };
char* hello = "3";

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

// Forcefully attaching socket to the port 8080
if (setsockopt(server_fd, 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 8080
if (bind(server_fd, (struct sockaddr*)&address,
          sizeof(address))
    < 0) {
    perror("bind failed");
    exit(EXIT_FAILURE);
}
if (listen(server_fd, 3) < 0) {
    perror("listen");
    exit(EXIT_FAILURE);
}
if ((new_socket
     = accept(server_fd, (struct sockaddr*)&address,
               (socklen_t*)&addrlen))
    < 0) {
    perror("accept");
    exit(EXIT_FAILURE);
}
valread = 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_fd, SHUT_RDWR);
return 0;
}

```

## Compiling client.c

```
cnlab404@cnlab404-Veriton-M200-H110:~$ gcc client.c -o client
cnlab404@cnlab404-Veriton-M200-H110:~$ ./client
```

Now compiling server.c and Hello message is sent . The number 3 which is sent by client is Checked in server.c and output came which is odd number .

```
cnlab404@cnlab404-Veriton-M200-H110:~$ nano server.c
cnlab404@cnlab404-Veriton-M200-H110:~$ gcc server.c -o server
cnlab404@cnlab404-Veriton-M200-H110:~$ ./server
Hello message sent
3
3number is odd
cnlab404@cnlab404-Veriton-M200-H110:~$
```

server.c file :\_

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

```
void oddeven(int n){}
int main(int argc, char const* argv[])
{
    int status, valread, client_fd;
    struct sockaddr_in serv_addr;
    char* hello = " hello from client ";
    char buffer[1024] = { 0 };

    if ((client_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\n Socket creation error \n");
        return -1;
    }

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

    // Convert IPv4 and IPv6 addresses from text to binary
```

```

// form
if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)
    <= 0) {
    printf(
        "\nInvalid address/ Address not supported \n");
    return -1;
}

if ((status
    = connect(client_fd, (struct sockaddr*)&serv_addr,
        sizeof(serv_addr)))
    < 0) {
    printf("\nConnection Failed \n");
    return -1;
}
valread = read(client_fd, buffer, 1024);
printf("%s\n", buffer);
int y = atoi( buffer);
printf("%d",y);
if(y%2 == 0){
    send(client_fd, "number is even in client", strlen("number is even
in client"), 0);
printf("number is even\n");}
else {
    send(client_fd, "number is odd in client", strlen("number is odd
in client"), 0);
printf("number is odd\n");}

    // closing the connected socket
    close(client_fd);

    return 0;
}

```

In this slide a message is sent to server by client i.e number 3 is sent by client to server to check whether 3 is odd or even and after that no. is checked in server.c and after checking it again sent to client.c and client.c displayed it .

```

cnlab404@cnlab404-Veriton-M200-H110:~$ gcc client.c -o client
cnlab404@cnlab404-Veriton-M200-H110:~$ ./client
hello from client
Hello message sent
cnlab404@cnlab404-Veriton-M200-H110:~$ gcc client.c -o client
cnlab404@cnlab404-Veriton-M200-H110:~$ ./client
hello from client
Hello message sent
cnlab404@cnlab404-Veriton-M200-H110:~$ gcc client.c -o client
cnlab404@cnlab404-Veriton-M200-H110:~$ ./client

^Z
[1]+  Stopped                  ./client
cnlab404@cnlab404-Veriton-M200-H110:~$ nano client.c
cnlab404@cnlab404-Veriton-M200-H110:~$ gcc client.c -o client
cnlab404@cnlab404-Veriton-M200-H110:~$ ./client

^Z
[2]+  Stopped                  ./client
cnlab404@cnlab404-Veriton-M200-H110:~$ nano client.c
cnlab404@cnlab404-Veriton-M200-H110:~$ gcc client.c -o client
cnlab404@cnlab404-Veriton-M200-H110:~$ ./client
Hello message sent
number is odd in client
cnlab404@cnlab404-Veriton-M200-H110:~$

```

Conclusion :- In this experiment I understand the concept of socket programming and understand how client and server communicate with each other .