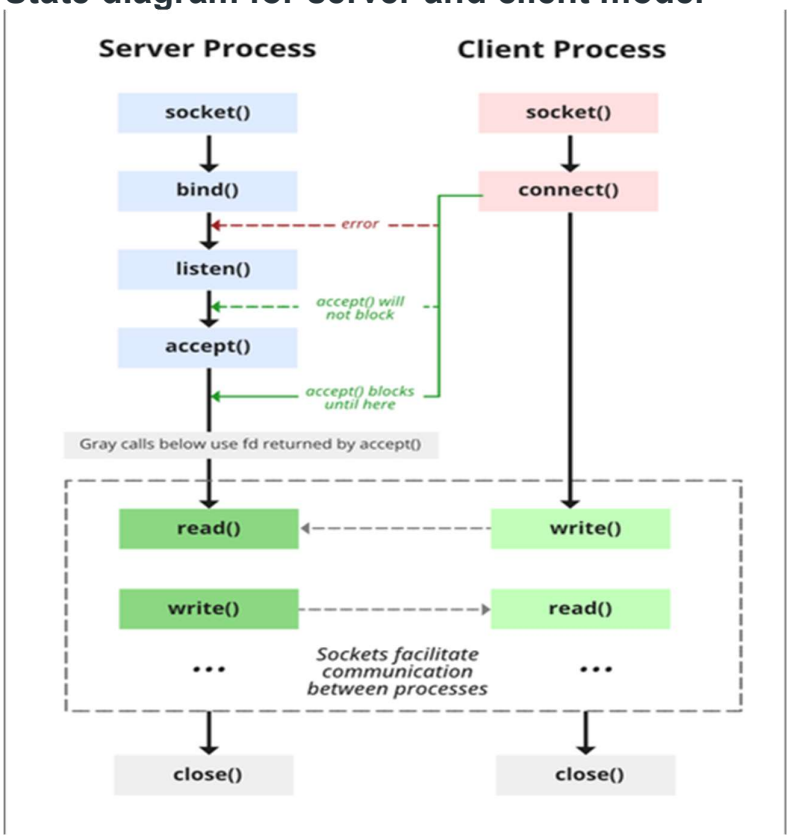
NAME – ATUL SHARMA
UID – 2021700060
BATCH – D4
Date :- 3/3/23

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

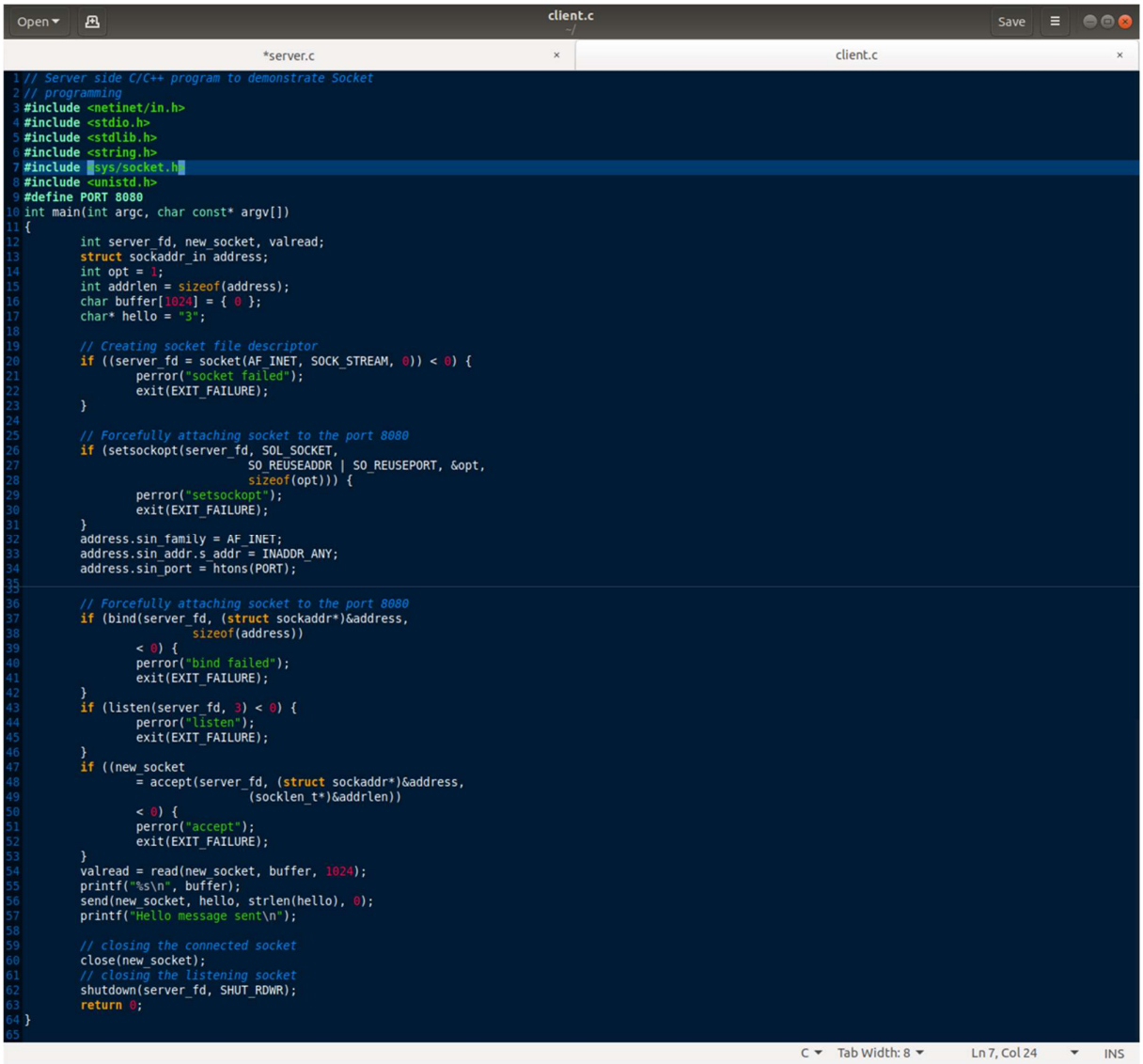
```c
// Client side C/C++ program to demonstrate Socket
// programming
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>
#define PORT 8080
void  oddeven(int n){
if(n%2 == 0){
printf("number is even\n");}
else {
printf("number 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 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



/ 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 .