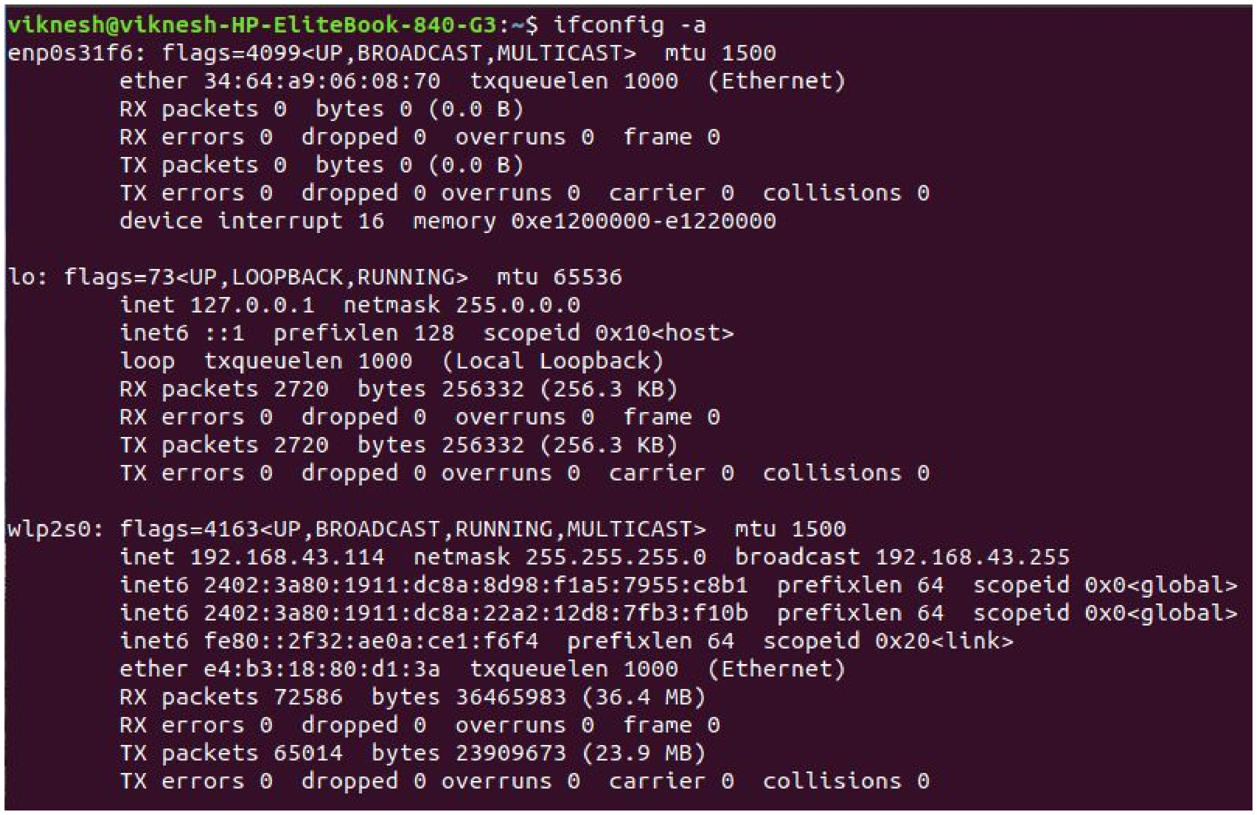
COMPUTER NETWORKS PRACTICE **VIKNESH RAJARAMON**

ASSIGNMENT 1 **COE18B060**

Linux Commands

ifconfig:

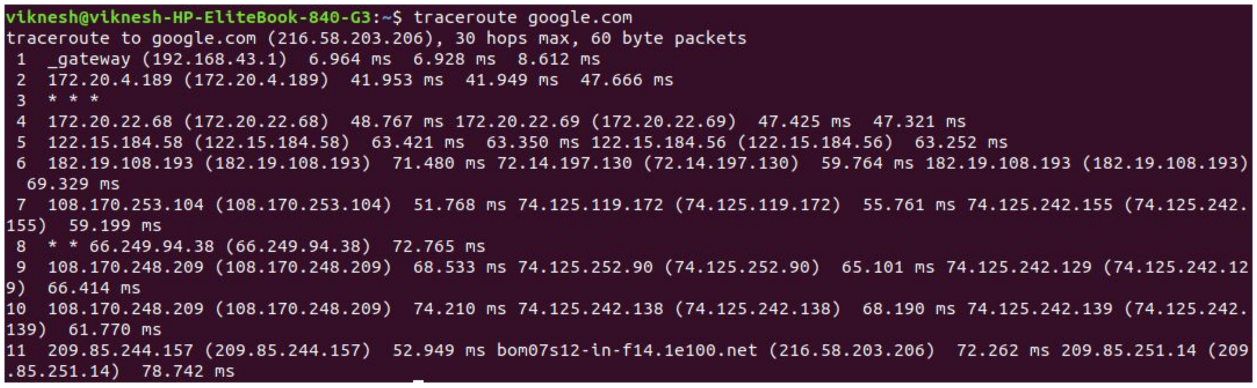
Used to configure the network interfaces linked with the kernel modules. This command allows us to assign an IP address, enable or disable a given interface. The “-a” option allows us to display all the interfaces.



```
viknesh@viknesh-HP-EliteBook-840-G3:~$ ifconfig -a
enp0s31f6: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 34:64:a9:06:08:70  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xe1200000-e1220000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2720  bytes 256332 (256.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2720  bytes 256332 (256.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.114  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 2402:3a80:1911:dc8a:8d98:f1a5:7955:c8b1  prefixlen 64  scopeid 0x0<global>
        inet6 2402:3a80:1911:dc8a:22a2:12d8:7fb3:f10b  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::2f32:ae0a:ce1:f6f4  prefixlen 64  scopeid 0x20<link>
        ether e4:b3:18:80:d1:3a  txqueuelen 1000  (Ethernet)
        RX packets 72586  bytes 36465983 (36.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 65014  bytes 23909673 (23.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

traceroute :

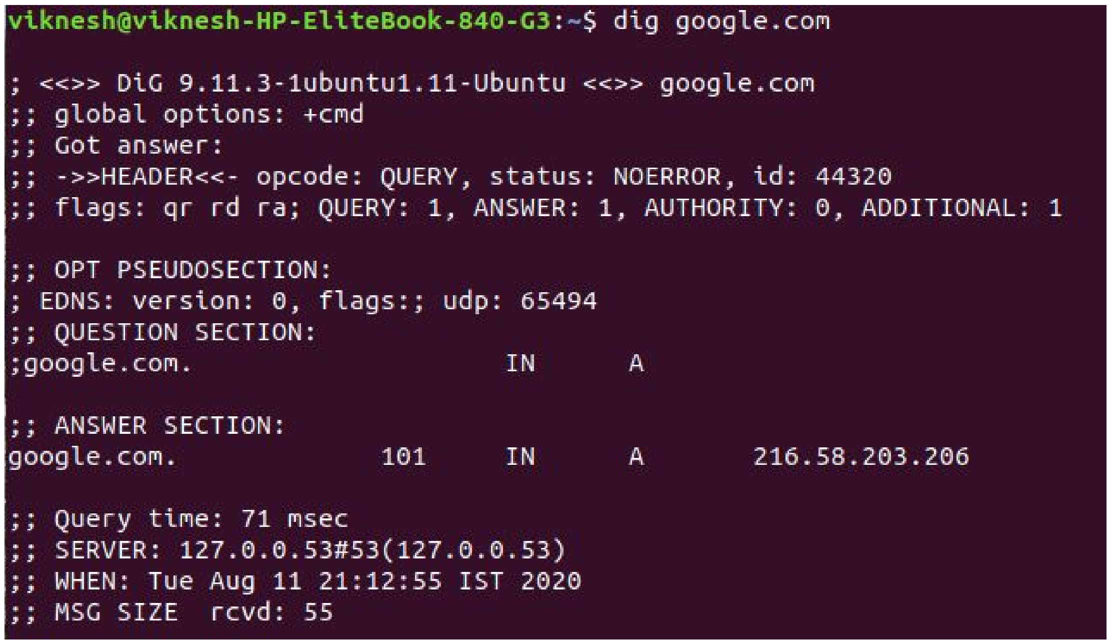
Displays the route taken by a packet to reach the host.



```
viknesh@viknesh-HP-EliteBook-840-G3:~$ traceroute google.com
traceroute to google.com (216.58.203.206), 30 hops max, 60 byte packets
 1  _gateway (192.168.43.1)  6.964 ms  6.928 ms  8.612 ms
 2  172.20.4.189 (172.20.4.189)  41.953 ms  41.949 ms  47.666 ms
 3  * * *
 4  172.20.22.68 (172.20.22.68)  48.767 ms 172.20.22.69 (172.20.22.69)  47.425 ms  47.321 ms
 5  122.15.184.58 (122.15.184.58)  63.421 ms  63.350 ms 122.15.184.56 (122.15.184.56)  63.252 ms
 6  182.19.108.193 (182.19.108.193)  71.480 ms 72.14.197.130 (72.14.197.130)  59.764 ms 182.19.108.193 (182.19.108.193)
  69.329 ms
 7  108.170.253.104 (108.170.253.104)  51.768 ms 74.125.119.172 (74.125.119.172)  55.761 ms 74.125.242.155 (74.125.242.
155)  59.199 ms
 8  * * 66.249.94.38 (66.249.94.38)  72.765 ms
 9  108.170.248.209 (108.170.248.209)  68.533 ms 74.125.252.90 (74.125.252.90)  65.101 ms 74.125.242.129 (74.125.242.12
9)  66.414 ms
10  108.170.248.209 (108.170.248.209)  74.210 ms 74.125.242.138 (74.125.242.138)  68.190 ms 74.125.242.139 (74.125.242.
139)  61.770 ms
11  209.85.244.157 (209.85.244.157)  52.949 ms bom07s12-in-f14.1e100.net (216.58.203.206)  72.262 ms 209.85.251.14 (209
.85.251.14)  78.742 ms
```

dig :

Stands for **Domain Information Groper**. It retrieves information about DNS name servers and is used for verifying and troubleshooting DNS problems.



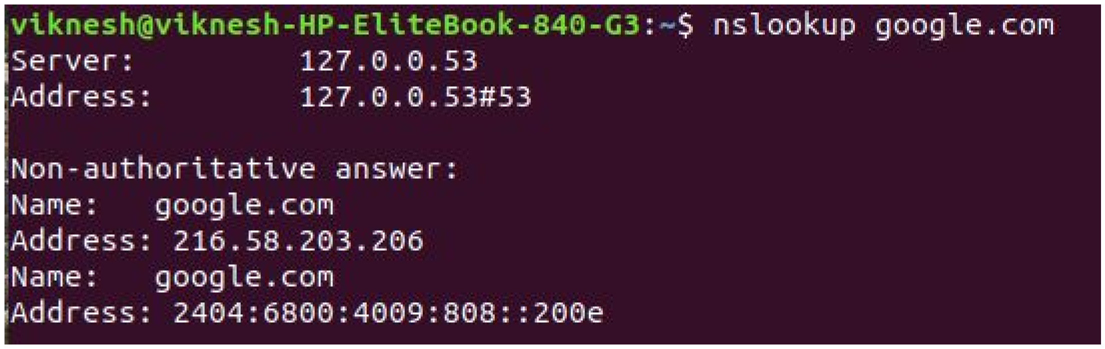
```
viknesh@viknesh-HP-EliteBook-840-G3:~$ dig google.com

; <<>> DiG 9.11.3-1ubuntu1.11-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44320
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             101     IN      A       216.58.203.206

;; Query time: 71 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Tue Aug 11 21:12:55 IST 2020
;; MSG SIZE  rcvd: 55
```

telnet :

Connect destination host:port via a TELNET protocol if connection establishes means connectivity between two hosts is working fine.

nslookup:

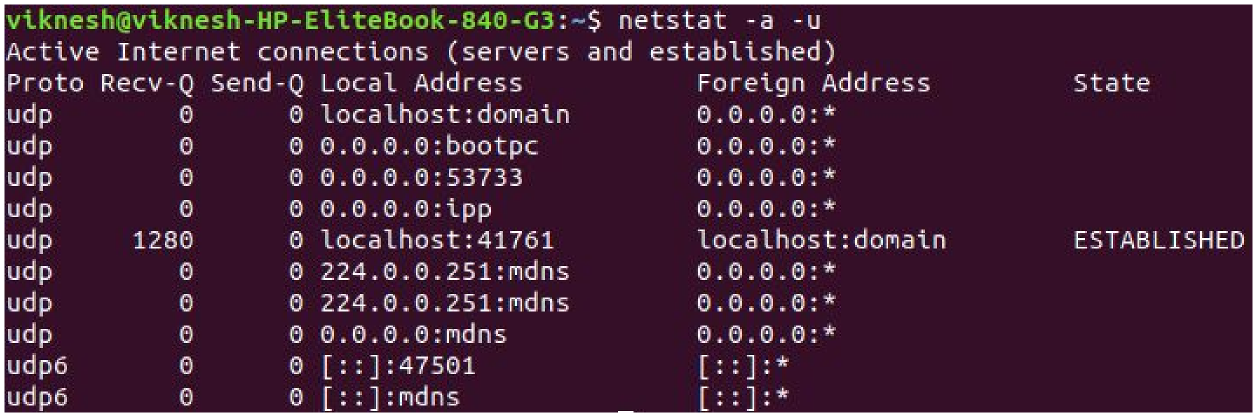
Used for querying the DNS to obtain domain name or IP address or other DNS records.



```
viknesh@viknesh-HP-EliteBook-840-G3:~$ nslookup google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 216.58.203.206
Name:   google.com
Address: 2404:6800:4009:808::200e
```

netstat :

Displays various network related information such as network connections, routing tables, interface statistics, etc.



```
viknesh@viknesh-HP-EliteBook-840-G3:~$ netstat -a -u
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 localhost:domain        0.0.0.0:*
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
udp        0      0 0.0.0.0:53733           0.0.0.0:*
udp        0      0 0.0.0.0:ipp             0.0.0.0:*
udp     1280      0 localhost:41761         localhost:domain        ESTABLISHED
udp        0      0 224.0.0.251:mdns        0.0.0.0:*
udp        0      0 224.0.0.251:mdns        0.0.0.0:*
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp6       0      0 [::]:47501              [::]:*
udp6       0      0 [::]:mdns               [::]:*
```

Socket API

int socket(int domain,int type,int protocol) :

domain - AF_INET(for IPv4 protocol) **(OR)** AF_INET6(for IPv6 protocol)

type - SOCK_DGRAM(for UDP) **(OR)** SOCK_STREAM(for TCP)

protocol - protocol value for IP which is 0.

On successful completion, it returns a non-negative value as the socket file descriptor. Otherwise, it returns a value of -1.

int bind(int socket,const struct sockaddr *address, socklen_t addrlen) :

socket - file descriptor of the socket to be bound

address - points to the sockaddr structure containing the address to be bound to the socket

addrlen - length of the sockaddr structure pointed to by the address argument

On successful completion, it returns 0. Otherwise, it returns a value of -1.

int recvfrom(int socket,const void *buffer,size_t length,int flags, struct sockaddr* address,socklen_t addrlen) :

socket - socket file descriptor

buffer - points to the buffer where the message received must be stored

length - length of the buffer pointed to by the buffer argument (in bytes)

flags - type of message reception (0 in this program)

address - points to the sockaddr structure in which the sending address is to be stored

addrlen - length of the sockaddr structure pointed to by the address argument

On successful completion, it returns the length of the message received in bytes.
Otherwise, it returns a value of -1.

int sendto(int socket,const void *message,size_t length,int flags, struct sockaddr* address,socklen_t addrlen) :

socket - socket file descriptor

message - points to the buffer containing the message to be sent

length - size of the message to be sent (in bytes)

flags - type of message transmission (0 in this program)

address - points to the sockaddr structure containing the destination address addrlen - length of the sockaddr structure pointed to by the address argument

On successful completion, it returns the length of the message sent in bytes. Otherwise, it returns a value of -1.

close(int socket) :

socket - socket file descriptor to be closed

On successful completion, it returns 0. Otherwise, it returns a value of -1.

server.c

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

#define PORT 8000
#define MAXLINE 1024

int main()
{
    int sockfd;//Socket Descriptor
    char reply[MAXLINE];
    memset(reply,'\0',MAXLINE);
    char message[MAXLINE];
    memset(message,0,MAXLINE);
    struct sockaddr_in ClientAddr;
    int CLen=sizeof(ClientAddr);

    /*Creating a Socket*/
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd<0)
    {
```

```
		perror("Socket Creation failed...\n");
		return 0;
	}

	/*Assign the values for Client Address Structure*/
	memset((char *) &ClientAddr,0,sizeof(ClientAddr));
	ClientAddr.sin_family=AF_INET;
	ClientAddr.sin_port=htons(PORT);
	ClientAddr.sin_addr.s_addr=htonl(INADDR_ANY);

	/*Bind Socket to Port*/
	int b=bind(sockfd,(struct sockaddr *) &ClientAddr,sizeof(ClientAddr));
	if(b==-1)
	{
		perror("Socket not Binded to Port...\n");
		exit(1);
	}

	printf("\nEnter \"close client\" to close the client\n\n");

	while(1)
	{
		/*Receive message from Client*/
		int received=recvfrom(sockfd,message,MAXLINE,0,(struct sockaddr *) &ClientAddr,&CLen);
		if(received==-1)
		{
			perror("recvfrom() failed...\n");
			exit(1);
		}

		/*Print received message*/
		printf("Received Message : %s\n",message);

		memset(reply,0,MAXLINE);

		/*Ask Server to Enter a Message*/
		printf("\nEnter message : ");
		gets(reply);


		/*Send message to Client*/
		int sent=sendto(sockfd,reply,strlen(reply),0,(struct sockaddr *) &ClientAddr,CLen);
		if(sent==-1)
		{
			perror("sendto() failed...\n");
			exit(1);
		}
```

```
            memset(message,0,MAXLINE);
        }
}
```

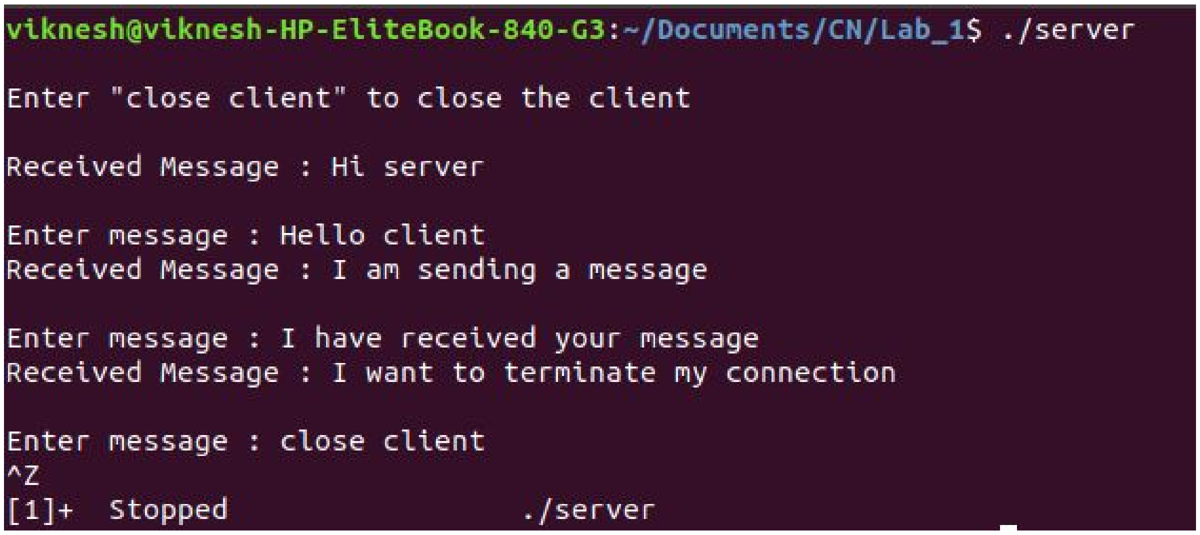
client.c

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

#define PORT 8000
#define MAXLINE 1024

int main()
{
    int sockfd;//Socket Descriptor
    char message[MAXLINE];
    memset(message,0,MAXLINE);
    char reply[MAXLINE];
    memset(reply,0,MAXLINE);
    struct sockaddr_in ServerAddr;
    int SLen=sizeof(ServerAddr);

    /*Creating a Socket*/
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd<0)
    {
            perror("Socket Creation failed...\n");
            return 0;
    }

    /*Assign the values for Server Address Structure*/
    ServerAddr.sin_family=AF_INET;
    ServerAddr.sin_port=htons(PORT);
    ServerAddr.sin_addr.s_addr=htonl(INADDR_ANY);
    memset(ServerAddr.sin_zero,0,sizeof(ServerAddr));
```

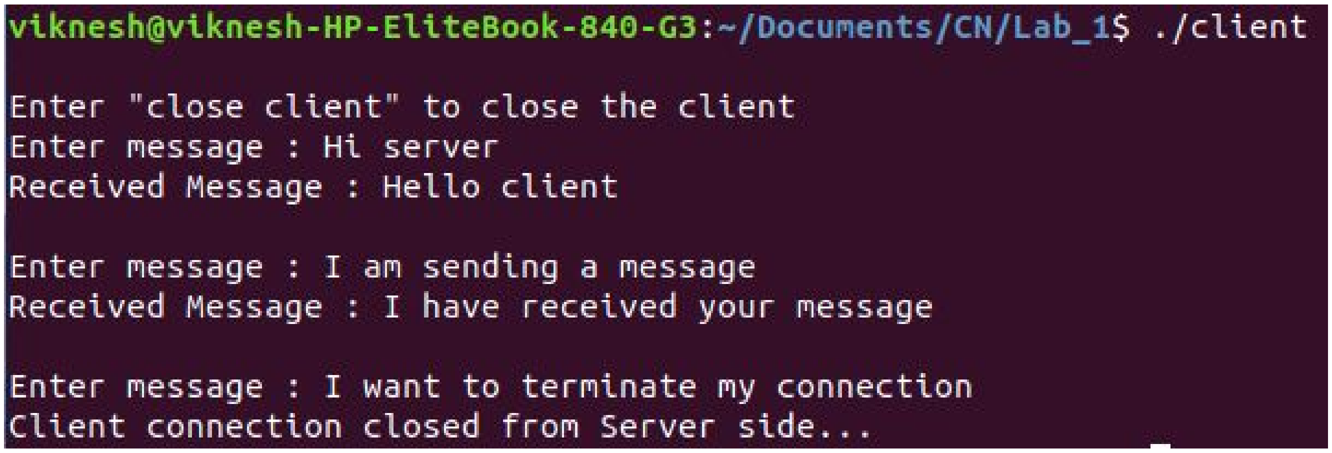
```
printf("\nEnter \"close client\" to close the client");

/*Ask Client to Enter a Message*/
printf("\nEnter message : ");
gets(message);

while((strcasecmp(message,"close client")!=0) && (strcasecmp(reply,"close client")!=0))
{
	/*Send message to Server*/
	int sent=sendto(sockfd,message,strlen(message),0,(struct sockaddr *) &ServerAddr,SLen);
	if(sent==-1)
	{
		perror("sendto() failed...\n");
		exit(1);
	}

	memset(reply,0,MAXLINE);

	/*Receive message from Server*/
	int received=recvfrom(sockfd,reply,MAXLINE,0,(struct sockaddr *) &ServerAddr,&SLen);
	if(received==-1)
	{
		perror("recvfrom() failed...\n");
		exit(1);
	}

	if(strcasecmp(reply,"close client")==0)
	{
		break;
	}

	/*Print received message*/
	printf("Received Message : %s\n",reply);

	memset(message,0,MAXLINE);

	/*Ask Client to Enter a Message*/
	printf("\nEnter message : ");
	gets(message);
}

if((strcasecmp(message,"close client")==0))
{
```

```
            printf("Client connection closed from Client side...\n");
    }
    else
    {
            printf("Client connection closed from Server side...\n");
    }

    /*Close the socket*/
    close(sockfd);
    return 0;
}
```

Output



```
viknesh@viknesh-HP-EliteBook-840-G3:~/Documents/CN/Lab_1$ ./server

Enter "close client" to close the client

Received Message : Hi server

Enter message : Hello client
Received Message : I am sending a message

Enter message : I have received your message
Received Message : I want to terminate my connection

Enter message : close client
^Z
[1]+  Stopped                 ./server
```



```
viknesh@viknesh-HP-EliteBook-840-G3:~/Documents/CN/Lab_1$ ./client

Enter "close client" to close the client
Enter message : Hi server
Received Message : Hello client

Enter message : I am sending a message
Received Message : I have received your message

Enter message : I want to terminate my connection
Client connection closed from Server side...
```