1. Implementation of Chat Server using TCP

Source Code:

TCP Client:

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
int main()

{
        int sock, bytes_recieved;
        char send_data[1024],recv_data[1024];

        struct sockaddr_in server_addr;
       sock = socket(AF_INET, SOCK_STREAM, 0);
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(9059);
        server_addr.sin_addr =htonl(INADDR_ANY);
        bzero(&(server_addr.sin_zero),8);
        connect(sock, (struct sockaddr *)&server_addr,
                    sizeof(struct sockaddr));
while(1)
        {
          printf("\nSEND (q or Q to quit) : ");
           scanf("%s",send_data);

          if (strcmp(send_data , "q") != 0 && strcmp(send_data , "Q") != 0)
           send(sock,send_data,strlen(send_data), 0);
          else
          {
           send(sock,send_data,strlen(send_data), 0);
           close(sock);
           break;
          }
        bytes_recieved=recv(sock,recv_data,1024,0);
        recv_data[bytes_recieved] = '\0';

        if (strcmp(recv_data , "q") == 0 || strcmp(recv_data , "Q") == 0)
             {
                close(sock);
                exit;
             }
         else
           printf("\nRecieved data = %s " , recv_data);

        }
 return 0;
 }
```

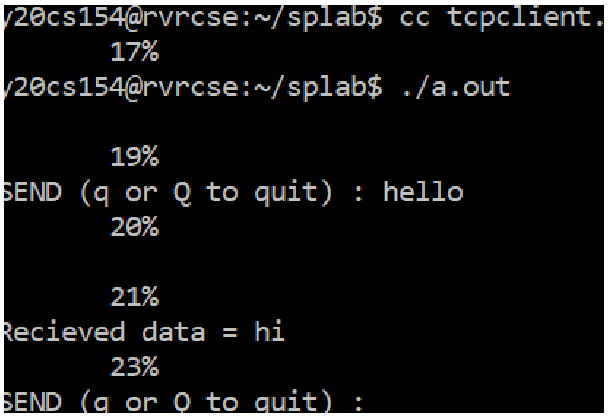
TCPServer:

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



int main()
{
        int sock, connected, bytes_recieved , true = 1;
        char send_data [1024] , recv_data[1024];

        struct sockaddr_in server_addr,client_addr;
        int sin_size;

        sock = socket(AF_INET, SOCK_STREAM, 0);
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(9059);
        server_addr.sin_addr.s_addr = INADDR_ANY;
        bind(sock, (struct sockaddr *)&server_addr, sizeof(struct sockaddr));
        listen(sock, 5);
        printf("\nTCPServer Waiting for client ");
```

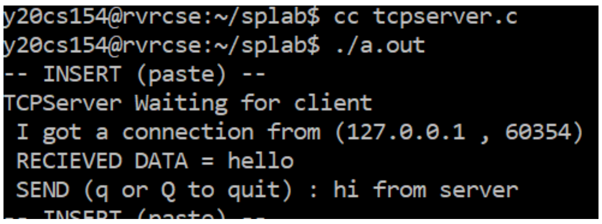
```
 fflush(stdout);
while(1)
 {

    sin_size = sizeof(struct sockaddr_in);

    connected = accept(sock, (struct sockaddr *)&client_addr,&sin_size);

    printf("\n I got a connection from (%s , %d)",
           inet_ntoa(client_addr.sin_addr),ntohs(client_addr.sin_port));

    while (1)
    {
      bytes_recieved = recv(connected,recv_data,1024,0);
      recv_data[bytes_recieved] = '\0';
      if (strcmp(recv_data , "q") == 0 || strcmp(recv_data , "Q") == 0)
      {
        close(connected);
        break;
      }
      else
      printf("\n RECIEVED DATA = %s " , recv_data);
        //sending data
       printf("\n SEND (q or Q to quit) : ");
      scanf("%s",send_data);
      if (strcmp(send_data , "q") == 0 || strcmp(send_data , "Q") == 0)
```

```
          {
            send(connected, send_data,strlen(send_data), 0);
            close(connected);
            break;
          }
          else
            send(connected, send_data,strlen(send_data), 0);

            fflush(stdout);
        }
      }

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

Output:



```
y20cs154@rvrcse:~/splab$ cc tcpclient.
       17%
y20cs154@rvrcse:~/splab$ ./a.out

       19%
SEND (q or Q to quit) : hello
       20%

       21%
Recieved data = hi
       23%
SEND (q or Q to quit) :
```



```
y20cs154@rvrcse:~/splab$ cc tcpserver.c
y20cs154@rvrcse:~/splab$ ./a.out
-- INSERT (paste) --
TCPServer Waiting for client
 I got a connection from (127.0.0.1 , 60354)
 RECIEVED DATA = hello
 SEND (q or Q to quit) : hi from server
```

2. Implementation of Daytime Server using TCP.

Source Code:

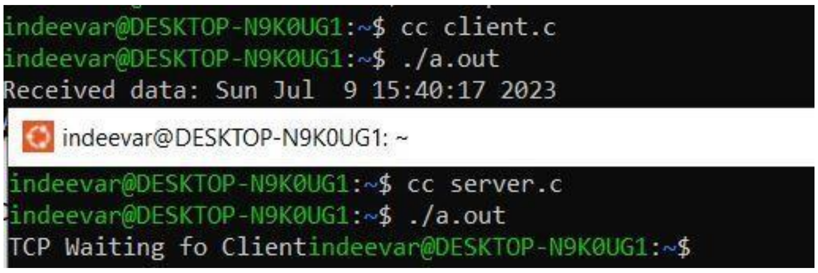
DayTimeClient:

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
int main()
{
	int sock,bytes;
	char recv_data[1024];
	struct sockaddr_in server_addr;
	sock=socket(AF_INET,SOCK_STREAM,0);
	server_addr.sin_family=AF_INET;
	server_addr.sin_port=htons(8523);
	server_addr.sin_addr.s_addr=htonl(INADDR_ANY);
	connect(sock,(struct sockaddr*)&server_addr,sizeof(struct sockaddr));
	bytes=recv(sock,recv_data,1024,0);
	printf("Received data: %s",recv_data);
	close(sock);
	return 0;
}
```

Daytime Server:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
#include<time.h>
int main()
{
        int sock,connected,bytes;
        struct sockaddr_in server_addr,client_addr;
        int sin_size;
        sock=socket(AF_INET,SOCK_STREAM,0);
        server_addr.sin_family=AF_INET;
        server_addr.sin_port=htons(8523);
        server_addr.sin_addr.s_addr=INADDR_ANY;
        bind(sock,(struct sockaddr*)&server_addr,sizeof(struct sockaddr));
        listen(sock,5);
        printf("TCP Waiting fo Client");
        sin_size=sizeof(struct sockaddr_in);
        connected=accept(sock,(struct sockaddr*)&client_addr,&sin_size);
        time_t t;
        time(&t);
        send(connected,ctime(&t),strlen(ctime(&t)),0);

        //close(connected);

        close(connected);

        close(sock);

        return 0;

}
```

Output:



```
indeevar@DESKTOP-N9K0UG1:~$ cc client.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Received data: Sun Jul  9 15:40:17 2023
```

indeevar@DESKTOP-N9K0UG1: ~

```
indeevar@DESKTOP-N9K0UG1:~$ cc server.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
TCP Waiting fo Clientindeevar@DESKTOP-N9K0UG1:~$
```

3. Implementation of Concurrent Echo Server using TCP.

SourceCode:

Echo Client:

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
int main()

{

    int sock, bytes_recieved;
    char send_data[1024],recv_data[1024];
    struct sockaddr_in server_addr;
   sock = socket(AF_INET, SOCK_STREAM, 0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(9934);
    server_addr.sin_addr.s_addr =htonl(INADDR_ANY);
    connect(sock, (struct sockaddr *)&server_addr,
          sizeof(struct sockaddr));
    while(1)
```

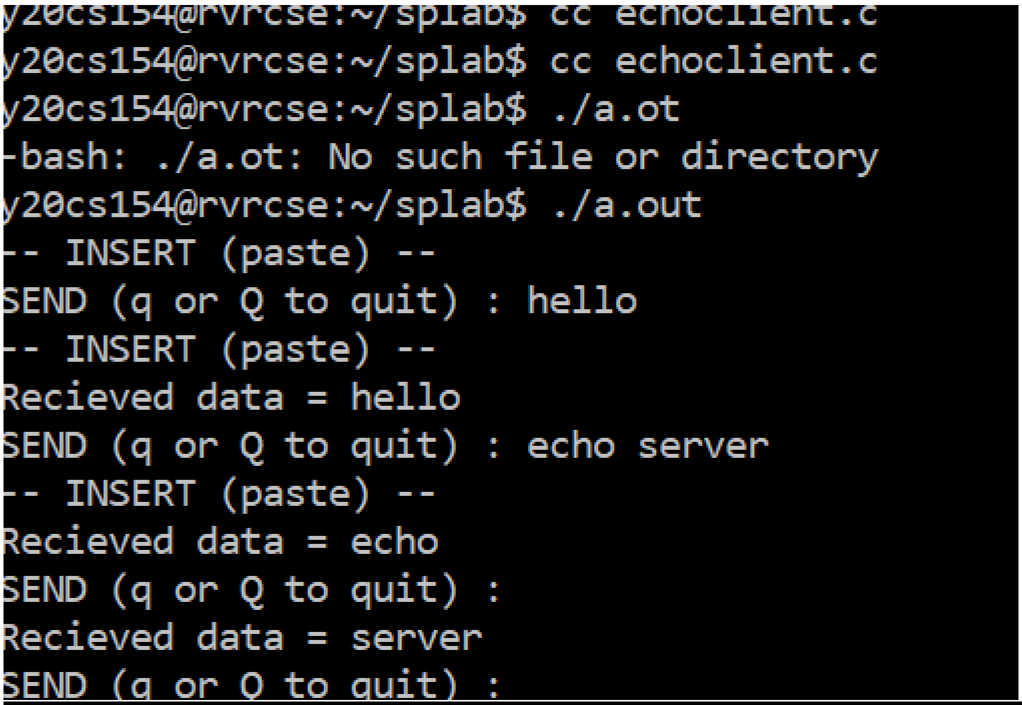
```
        {
          printf("\nSEND (q or Q to quit) : ");
           scanf("%s",send_data);

          if (strcmp(send_data , "q") != 0 && strcmp(send_data , "Q") != 0)
           send(sock,send_data,strlen(send_data), 0);

          else
          {
           send(sock,send_data,strlen(send_data), 0);
           close(sock);
           break;
          }
        bytes_recieved=recv(sock,recv_data,1024,0);
        recv_data[bytes_recieved] = '\0';

        if (strcmp(recv_data , "q") == 0 || strcmp(recv_data , "Q") == 0)
            {
              close(sock);
              exit;
            }
        else
           printf("\nRecieved data = %s " , recv_data);

        }
    return 0;
    }
```

Echo Server:

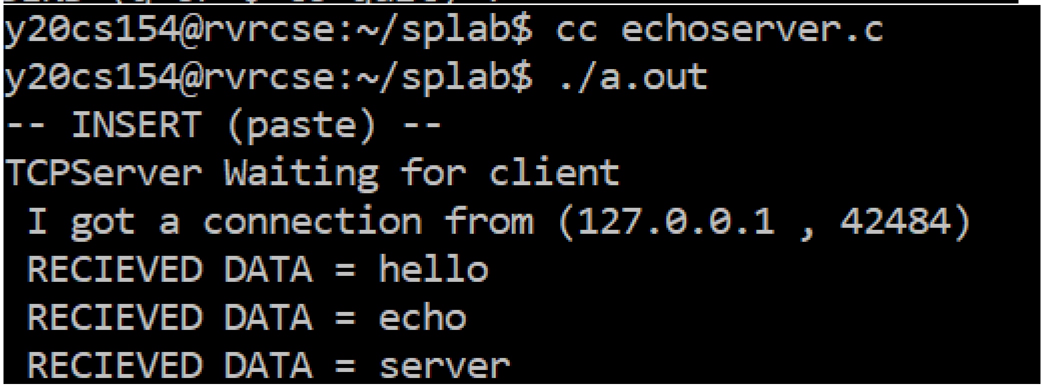
```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>


int main()
{
        int sock, connected, bytes_recieved , true = 1;
        char send_data [1024] , recv_data[1024];

        struct sockaddr_in server_addr,client_addr;
        int sin_size;
        int pid;
        sock = socket(AF_INET, SOCK_STREAM, 0);
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(9934);
        server_addr.sin_addr.s_addr = INADDR_ANY;
        bind(sock, (struct sockaddr *)&server_addr, sizeof(struct sockaddr));
        listen(sock, 5);
        printf("\nTCPServer Waiting for client ");
        fflush(stdout);
       while(1)
        {

            sin_size = sizeof(struct sockaddr_in);

            connected = accept(sock, (struct sockaddr *)&client_addr,&sin_size);
         printf("\n I got a connection from (%s , %d)",
                   inet_ntoa(client_addr.sin_addr),ntohs(client_addr.sin_port));
           if((pid=fork())==0)
```

```
          {
                close(sock);
           while (1)
           {
             bytes_recieved = recv(connected,recv_data,1024,0);
             recv_data[bytes_recieved] = '\0';
             if (strcmp(recv_data , "q") == 0 || strcmp(recv_data , "Q") == 0)
             {
               close(connected);
               break;
             }
             else
             printf("\n RECIEVED DATA = %s " , recv_data);
               //sending data
              //printf("\n SEND (q or Q to quit) : ");
            //scanf("%s",send_data);
             strcpy(send_data,recv_data);
             if (strcmp(send_data , "q") == 0 || strcmp(send_data , "Q") == 0)
             {
               send(connected, send_data,strlen(send_data), 0);
               close(connected);
               break;
             }
             else
                send(connected, send_data,strlen(send_data), 0);

               fflush(stdout);
           }
                exit(0);
       }
      close(connected);
       }

     return 0;
}
```

Output:



```
y20cs154@rvrcse:~/splab$ cc echoclient.c
y20cs154@rvrcse:~/splab$ cc echoclient.c
y20cs154@rvrcse:~/splab$ ./a.ot
-bash: ./a.ot: No such file or directory
y20cs154@rvrcse:~/splab$ ./a.out
-- INSERT (paste) --
SEND (q or Q to quit) : hello
-- INSERT (paste) --
Recieved data = hello
SEND (q or Q to quit) : echo server
-- INSERT (paste) --
Recieved data = echo
SEND (q or Q to quit) :
Recieved data = server
SEND (q or Q to quit) :
```



```
y20cs154@rvrcse:~/splab$ cc echoserver.c
y20cs154@rvrcse:~/splab$ ./a.out
-- INSERT (paste) --
TCPServer Waiting for client
 I got a connection from (127.0.0.1 , 42484)
 RECIEVED DATA = hello
 RECIEVED DATA = echo
 RECIEVED DATA = server
```

4. Implementation of Computational Server using TCP.

SourceCode:

CSClient:

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
int main()
{
    int sock,bytes;
    char send_data[1024],recv_data[1024];
    struct sockaddr_in server_addr;
    sock=socket(AF_INET,SOCK_STREAM,0);
    server_addr.sin_family=AF_INET;
    server_addr.sin_port=htons(8563);
    server_addr.sin_addr.s_addr=htonl(INADDR_ANY);
    connect(sock,(struct sockaddr *)&server_addr,sizeof(struct sockaddr));
    printf("Enter data\n");
    scanf("%s",send_data);
    send(sock,send_data,strlen(send_data),0);
    bytes=recv(sock,recv_data,1024,0);
```

```
        recv_data[bytes]='\0';
        close(sock);
        printf("Received Data %s",recv_data);
        return 0;
}
```

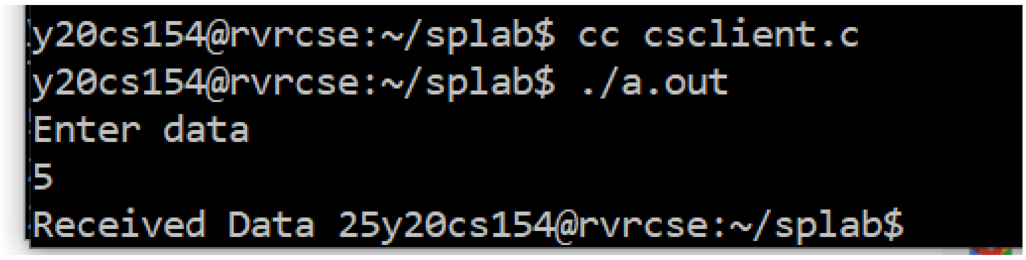
CSServer:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include<math.h>
int main()
{
        int sock,connected,bytes;
        char send_data[1024],recv_data[1024];
        struct sockaddr_in server_addr,client_addr;
        int sin_size;
        sock=socket(AF_INET,SOCK_STREAM,0);
        server_addr.sin_family=AF_INET;
        server_addr.sin_port=htons(8563);
        server_addr.sin_addr.s_addr=INADDR_ANY;
        bind(sock,(struct sockaddr *)&server_addr,sizeof(struct sockaddr));
```

```
listen(sock,5);
printf("TCPServer Waiting for client\n");
sin_size=sizeof(struct sockaddr_in);
connected=accept(sock,(struct sockaddr *)&client_addr,&sin_size);
bytes=recv(connected,recv_data,1024,0);
recv_data[bytes]='\0';
int i=0,k=0;
char ch1[9];
int k1=0;
while(recv_data[i]!='\0')
{
        ch1[k]=recv_data[i];
        k=k+1;
        i=i+1;
}
ch1[k]='\0';
int val=atoi(ch1);
printf("%s\n",ch1);
int res=val*val;
printf("%d\n",res);
while(res>0)
{
        int d=res%10;
        res=res/10;
        send_data[k1]=d+'0';
        k1=k1+1;
}
```

```
        send_data[k1]='\0';
        char send_data2[1024];
        int k2=0;
        for(i=k1-1;i>=0;i--)
        {
                send_data2[k2]=send_data[i];
                k2=k2+1;
        }
        send_data2[k2]='\0';
        printf("%s",send_data2);
        send(connected,send_data2,strlen(send_data2),0);
        close(connected);
        close(sock);
        return 0;
}
```

Output:



```
y20cs154@rvrcse:~/splab$ cc csclient.c
y20cs154@rvrcse:~/splab$ ./a.out
Enter data
5
Received Data 25y20cs154@rvrcse:~/splab$
```

5. Implementation of DNS Server using TCP.

SourceCode:

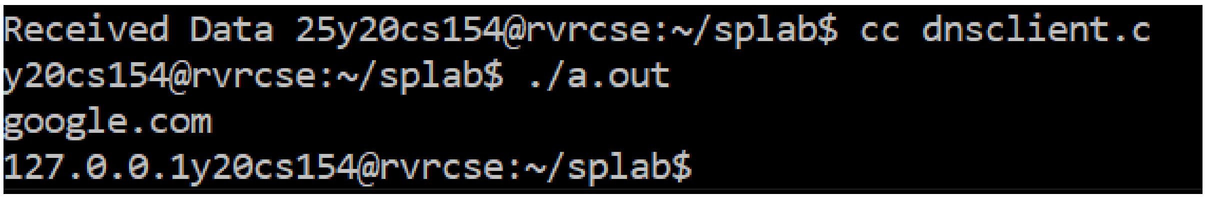
DNSClient:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
int main()
{
    int sock,bytes;
    char send_data[1024],recv_data[1024];
    struct sockaddr_in server_addr;
    sock=socket(AF_INET,SOCK_STREAM,0);
    server_addr.sin_family=AF_INET;
    server_addr.sin_port=htons(8563);
    server_addr.sin_addr.s_addr=htonl(INADDR_ANY);
    connect(sock,(struct sockaddr*)&server_addr,sizeof(struct sockaddr));
    scanf("%s",send_data);
    send(sock,send_data,strlen(send_data),0);
    bytes=recv(sock,recv_data,1024,0);
    recv_data[bytes]='\0';
    printf("%s",recv_data);
```

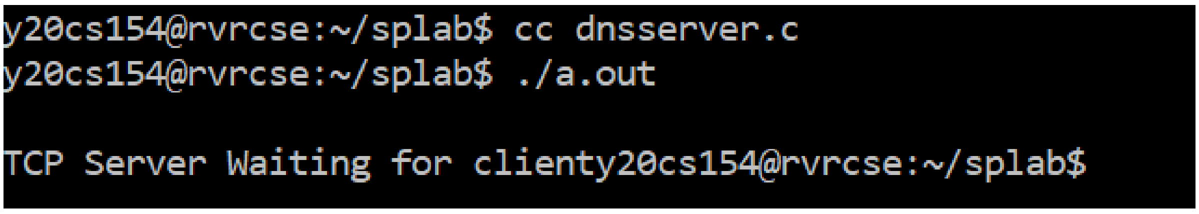
```
        return 0;
}

DNSServer:
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
#include<string.h>
int main()
{
        int sock,connected,bytes;
        char send_data[1024],recv_data[1024];
        struct sockaddr_in server_addr,client_addr;
        int sin_size;
        sock=socket(AF_INET,SOCK_STREAM,0);
        server_addr.sin_family=AF_INET;
        server_addr.sin_port=htons(8563);
        server_addr.sin_addr.s_addr=INADDR_ANY;
        bind(sock,(struct sockaddr *)&server_addr,sizeof(struct sockaddr));
        listen(sock,5);
        printf("\nTCP Server Waiting for client");
        sin_size=sizeof(struct sockaddr_in);
        connected=accept(sock,(struct sockaddr *)&client_addr,&sin_size);
```

```
bytes=recv(connected,recv_data,1024,0);
recv_data[bytes]='\0';
char dnsname[][100]={"google.com","facebook.com","amazon.com"};
char dnsip[][100]={"127.0.0.1","128.0.0.1","129.0.0.1"};
int i,index=-1;
for(i=0;i<3;i++)
{
        if(strcmp(dnsname[i],recv_data)==0)
        {
                index=i;
                break;
        }
}
if(index==-1)
{
        send(connected,"-1",1,0);
        close(connected);
        close(sock);
}
else
{
        strcpy(send_data,dnsip[index]);
}
send(connected,send_data,strlen(send_data),0);
close(connected);
close(sock);
return 0;}
```

Output:



```
Received Data 25y20cs154@rvrcse:~/splab$ cc dnsclient.c
y20cs154@rvrcse:~/splab$ ./a.out
google.com
127.0.0.1y20cs154@rvrcse:~/splab$
```



```
y20cs154@rvrcse:~/splab$ cc dnsserver.c
y20cs154@rvrcse:~/splab$ ./a.out

TCP Server Waiting for clienty20cs154@rvrcse:~/splab$
```

6. Implementation of Authentication Server using TCP.

SourceCode:

Authclient:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
int main()
{
	int sock,bytes;
	char send_data[1024],recv_data[1024];
	char password[1024];
	struct sockaddr_in server_addr;
	sock=socket(AF_INET,SOCK_STREAM,0);
	server_addr.sin_family=AF_INET;
	server_addr.sin_port=htons(8563);
	server_addr.sin_addr.s_addr=htonl(INADDR_ANY);
	connect(sock,(struct sockaddr*)&server_addr,sizeof(struct sockaddr));
	printf("Enter Username");
	scanf("%s",send_data);
	send(sock,send_data,strlen(send_data),0);
```

```
        printf("Enter Password");
        scanf("%s",password);
        send(sock,password,strlen(password),0);
        bytes=recv(sock,recv_data,1024,0);
        recv_data[bytes]='\0';
        printf("%s",recv_data);
        return 0;
}
```

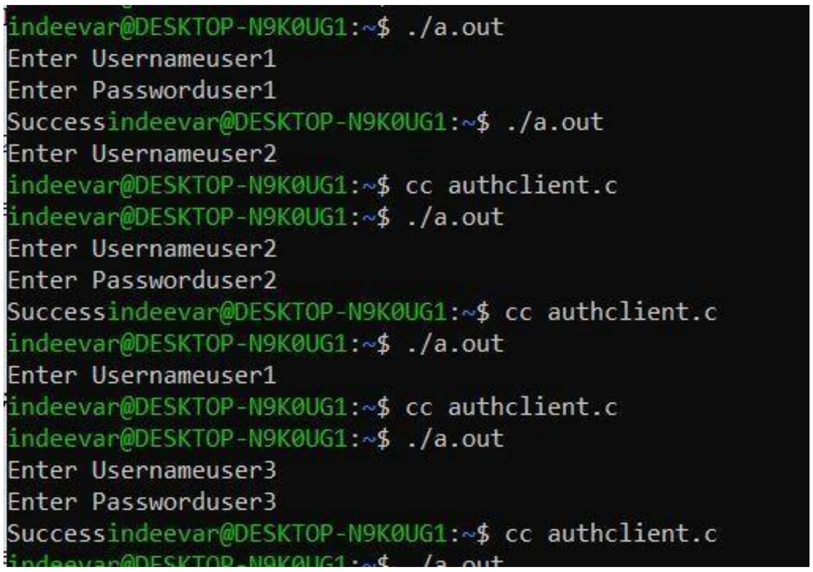
AuthServer:

```
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
#include<string.h>
int main()
{
	int sock,connected,bytes,bytes1;
	char send_data[1024],recv_data[1024];
	char pass[1024];
	struct sockaddr_in server_addr,client_addr;
	int sin_size;
	sock=socket(AF_INET,SOCK_STREAM,0);
	server_addr.sin_family=AF_INET;
	server_addr.sin_port=htons(8563);
	server_addr.sin_addr.s_addr=htonl(INADDR_ANY);
	bind(sock,(struct sockaddr *)&server_addr,sizeof(struct sockaddr));
	listen(sock,5);
	printf("\nTCP Server Waiting for client");
	while(1)
	{
	sin_size=sizeof(struct sockaddr_in);
	connected=accept(sock,(struct sockaddr *)&client_addr,&sin_size);
```

```
while(1)
{
bytes=recv(connected,recv_data,1024,0); recv_data[bytes]='\0';
if(strcmp(recv_data,"q")==0)
{
        close(connected);
        break;
}
bytes1=recv(connected,pass,1024,0);
pass[bytes1]='\0';
char username[][100]={"user1","user2","user3"};
char password[][100]={"user1","user2","user3"};
int i,index=-1;
for(i=0;i<3;i++)
{
        if(strcmp(username[i],recv_data)==0)
        {
                index=i;
                break;
        }
}
if(index==-1)
{
        strcpy(send_data,"Wrong User");
        send(connected,send_data,1024,0);
       // close(connected);
       // close(sock);
```

```
        }
        else
        {
                if(strcmp(pass,password[i])==0)             {
                        strcpy(send_data,"Success");
                }
                else
                {
                        strcpy(send_data,"Error");
                }
        }
        send(connected,send_data,strlen(send_data),0);
        //close(connected);
        //close(sock);
        }
        }
        return 0;
}
```

Output:



```
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Usernameuser1
Enter Passworduser1
Successindeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Usernameuser2
indeevar@DESKTOP-N9K0UG1:~$ cc authclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Usernameuser2
Enter Passworduser2
Successindeevar@DESKTOP-N9K0UG1:~$ cc authclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Usernameuser1
indeevar@DESKTOP-N9K0UG1:~$ cc authclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Usernameuser3
Enter Passworduser3
Successindeevar@DESKTOP-N9K0UG1:~$ cc authclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
```

7. Implementation of FTP Using TCP

SourceCode:

FTPClient:

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


int main()

{

    int sock, bytes_recieved;
    char send_data[1024],recv_data[1024];
    struct sockaddr_in server_addr;
   sock = socket(AF_INET, SOCK_STREAM, 0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(9934);
    server_addr.sin_addr.s_addr =htonl(INADDR_ANY);
    connect(sock, (struct sockaddr *)&server_addr,
                sizeof(struct sockaddr));
     /* printf("\nSEND (q or Q to quit) : ");
```

```
          scanf("%s",send_data);

         if (strcmp(send_data , "q") != 0 && strcmp(send_data , "Q") != 0)
          send(sock,send_data,strlen(send_data), 0);

         else
         {
          send(sock,send_data,strlen(send_data), 0);
          close(sock);
          break;
         }*/
        bytes_recieved=recv(sock,recv_data,1024,0);
        recv_data[bytes_recieved] = '\0';
        printf("\nRecieved data = %s " , recv_data);

return 0;
}
```

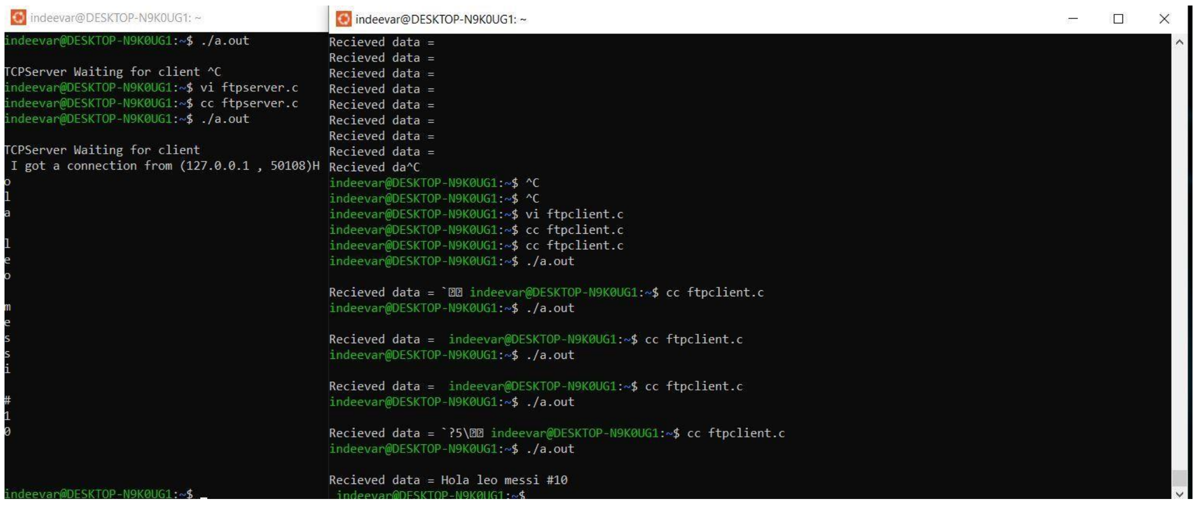
FTPServer:

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


int main()
{
        int sock, connected, bytes_recieved , true = 1;
        char send_data [1024] , recv_data[1024];

        struct sockaddr_in server_addr,client_addr;
        int sin_size;
        int pid;
        FILE *fp;
        char ch;
        sock = socket(AF_INET, SOCK_STREAM, 0);
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(9934);
        server_addr.sin_addr.s_addr = INADDR_ANY;
        bind(sock, (struct sockaddr *)&server_addr, sizeof(struct sockaddr));
        listen(sock, 5);
```

```
printf("\nTCPServer Waiting for client ");
fflush(stdout);
    sin_size = sizeof(struct sockaddr_in);
    connected = accept(sock, (struct sockaddr *)&client_addr,&sin_size);
 printf("\n I got a connection from (%s , %d)",
          inet_ntoa(client_addr.sin_addr),ntohs(client_addr.sin_port));
     //bytes_recieved = recv(connected,recv_data,1024,0);
     //recv_data[bytes_recieved] = '\0';
     /*if (strcmp(recv_data , "q") == 0 || strcmp(recv_data , "Q") == 0)
     {
       close(connected);
       break;
     }*/
    // else
    // printf("\n RECIEVED DATA = %s " , recv_data);
      //sending data
     //printf("\n SEND (q or Q to quit) : ");
    //scanf("%s",send_data);
    int k=0;
    fp=fopen("test.txt","r");
    do
    {
            ch=fgetc(fp);
            printf("%c\n",ch);
            send_data[k]=ch;
            k=k+1;
    }while(ch!=EOF);
```

```
            send_data[k]='\0';

            send(connected, send_data,strlen(send_data), 0);

              fflush(stdout);

        close(connected);


       return 0;

}
```

Output:



```
indeevar@DESKTOP-N9K0UG1:~$ ./a.out

TCPServer Waiting for client ^C
indeevar@DESKTOP-N9K0UG1:~$ vi ftpserver.c
indeevar@DESKTOP-N9K0UG1:~$ cc ftpserver.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out

TCPServer Waiting for client
 I got a connection from (127.0.0.1 , 50108)H
o
l
a

l
e
o

m
e
s
s
i

#
1
0



indeevar@DESKTOP-N9K0UG1:~$ _
```

```
Recieved data =
Recieved data =
Recieved data =
Recieved data =
Recieved data =
Recieved data =
Recieved data =
Recieved data =
Recieved da^C
indeevar@DESKTOP-N9K0UG1:~$ ^C
indeevar@DESKTOP-N9K0UG1:~$ ^C
indeevar@DESKTOP-N9K0UG1:~$ vi ftpclient.c
indeevar@DESKTOP-N9K0UG1:~$ cc ftpclient.c
indeevar@DESKTOP-N9K0UG1:~$ cc ftpclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out

Recieved data = `▯▯ indeevar@DESKTOP-N9K0UG1:~$ cc ftpclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out

Recieved data =  indeevar@DESKTOP-N9K0UG1:~$ cc ftpclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out

Recieved data =  indeevar@DESKTOP-N9K0UG1:~$ cc ftpclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out

Recieved data = `?5\▯▯ indeevar@DESKTOP-N9K0UG1:~$ cc ftpclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out

Recieved data = Hola leo messi #10
 indeevar@DESKTOP-N9K0UG1:~$
```

8. Implementation Of Caeser Cipher Using UDP

Soruce Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
#include<string.h>
#include<unistd.h>
int main()
{
        int sock,port,len,key=2;
        char
chars[26]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x',
'y','z'};
        struct sockaddr_in serveraddr;
        char send_data[1024],recv_data[1024];
        printf("Enter Port");
        scanf("%d",&port);
        sock=socket(AF_INET,SOCK_DGRAM,0);
        serveraddr.sin_family=AF_INET;
        serveraddr.sin_port=htons(port);
        serveraddr.sin_addr.s_addr=htonl(INADDR_ANY);
        //connect(sock,(struct sockaddr*)&serveraddr,sizeof(struct sockaddr));
        printf("Enter the String");
        scanf("%s",send_data);
        char caeser[1024];
        int i,j;
        for(i=0;i<strlen(send_data);i++)
        {
                for(j=0;j<26;j++)                    if(send_data[i]==chars[j])
                        {
                                int k=j+key;
                                if(k>=25)
                                {
                                        k=k%26;
                                }
                                caeser[i]=chars[k];
                        }
                }
        }
        caeser[strlen(send_data)]='\0';
        printf("Caeser Text %s",caeser);
        sendto(sock,caeser,strlen(caeser),0,(struct
sockaddr*)&serveraddr,sizeof(serveraddr));
        //int bytes;
        //bytes=recvfrom(sock,recv_data,1024,0,(struct sockaddr*)NULL,NULL);
        //recv_data[bytes]='\0';
        //printf("Plain Text is %s",recv_data);
        close(sock);
}
```

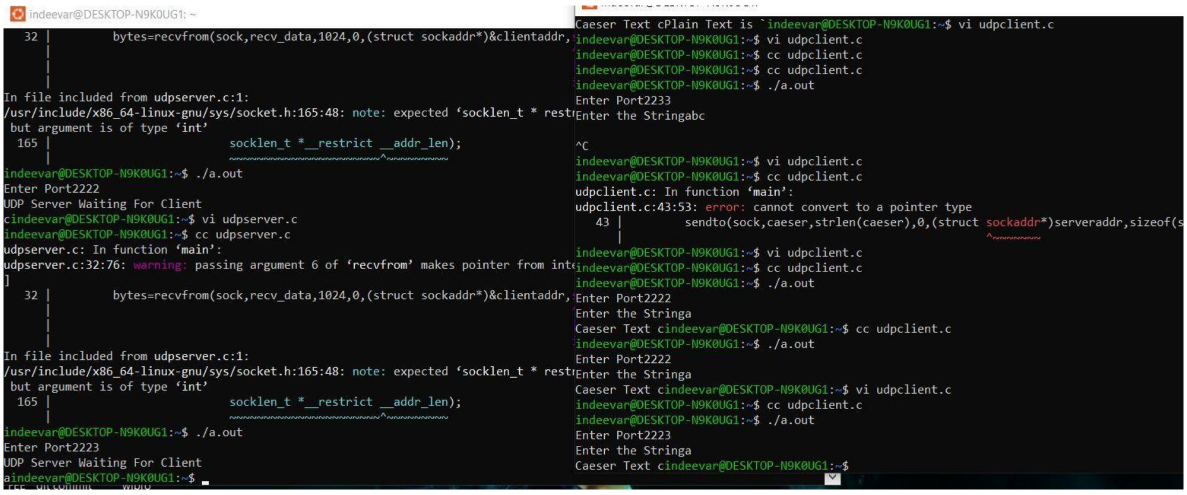
CaeserServer:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include<netdb.h>
int main()
{
	int sock,connect,bytes,port,len;
	struct sockaddr_in serveraddr,clientaddr;
	char
recv_data[1024],chars[26]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r',
's','t','u','v','w','x','y','z'};
	printf("Enter Port");
	scanf("%d",&port);
	int sin_size;
	sock=socket(AF_INET,SOCK_DGRAM,0);
	if(sock<0)
	{
		printf("Sock Error");
	}
	serveraddr.sin_family=AF_INET;
```

```
        serveraddr.sin_port=htons(port);
        serveraddr.sin_addr.s_addr=htonl(INADDR_ANY);
        bind(sock,(struct sockaddr *)&serveraddr,sizeof(struct sockaddr));
        listen(sock,5);
        printf("UDP Server Waiting For Client\n"); sin_size=sizeof(struct
sockaddr_in);
        //connect=accept(sock
        //(struct sockaddr *)&clientaddr,&sin_size);
        bytes=recvfrom(sock,recv_data,1024,0,(struct
sockaddr*)&clientaddr,sin_size);
        recv_data[bytes]='\0';
        char str[1024];
        int i,j;
        for(i=0;i<strlen(recv_data);i++)
        {
                for(j=0;j<26;j++)
                {
                        int key=2;
                        int k=j-key;
                        if(recv_data[i]==chars[j])
                        {
                        if(k<0)
                        {
                                k=k+26;
                        }
                        recv_data[i]=chars[k];
                        }
```

```
            }
        }
        recv_data[bytes]='\0';
        printf("%s",recv_data);
        sendto(sock,recv_data,strlen(recv_data),0,(struct
sockaddr*)&clientaddr,sin_size);
        close(connect);
        close(sock);
        //return 0;
}
```

Output:



```
indeevar@DESKTOP-N9K0UG1: ~
   32 |         bytes=recvfrom(sock,recv_data,1024,0,(struct sockaddr*)&clientaddr,
      |
      |
      |
In file included from udpserver.c:1:
/usr/include/x86_64-linux-gnu/sys/socket.h:165:48: note: expected ‘socklen_t * restr
 but argument is of type ‘int’
  165 |                           socklen_t *__restrict __addr_len);
      |                           ~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Port2222
UDP Server Waiting For Client
cindeevar@DESKTOP-N9K0UG1:~$ vi udpserver.c
indeevar@DESKTOP-N9K0UG1:~$ cc udpserver.c
udpserver.c: In function ‘main’:
udpserver.c:32:76: warning: passing argument 6 of ‘recvfrom’ makes pointer from inte
]
   32 |         bytes=recvfrom(sock,recv_data,1024,0,(struct sockaddr*)&clientaddr,
      |
      |
      |
In file included from udpserver.c:1:
/usr/include/x86_64-linux-gnu/sys/socket.h:165:48: note: expected ‘socklen_t * restr
 but argument is of type ‘int’
  165 |                           socklen_t *__restrict __addr_len);
      |                           ~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Port2223
UDP Server Waiting For Client
aindeevar@DESKTOP-N9K0UG1:~$
```

```
Caeser Text cPlain Text is `indeevar@DESKTOP-N9K0UG1:~$ vi udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ vi udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ cc udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ cc udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Port2233
Enter the Stringabc

^C
indeevar@DESKTOP-N9K0UG1:~$ vi udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ cc udpclient.c
udpclient.c: In function ‘main’:
udpclient.c:43:53: error: cannot convert to a pointer type
   43 |         sendto(sock,caeser,strlen(caeser),0,(struct sockaddr*)serveraddr,sizeof(s
      |                                                     ^~~~~~~~
indeevar@DESKTOP-N9K0UG1:~$ vi udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ cc udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Port2222
Enter the Stringa
Caeser Text cindeevar@DESKTOP-N9K0UG1:~$ cc udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Port2222
Enter the Stringa
Caeser Text cindeevar@DESKTOP-N9K0UG1:~$ vi udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ cc udpclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Port2223
Enter the Stringa
Caeser Text cindeevar@DESKTOP-N9K0UG1:~$
```

9. Implementation of daytime Server as a daemon.

SourceCode:

Daemonclient:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
int main()
{
    int sock,port,n;
    char  recv_data[1024+1];
    struct sockaddr_in serveraddr;
    printf("Enter Port");
    scanf("%d",&port);
    if((sock=socket(AF_INET,SOCK_STREAM,0))<0)
    {
        printf("Socket Error");
    }
    serveraddr.sin_family=AF_INET;
    serveraddr.sin_port=htons(port);
    serveraddr.sin_addr.s_addr=htonl(INADDR_ANY);
    if(connect(sock,(struct sockaddr*)&serveraddr,sizeof(struct sockaddr))<0)
```

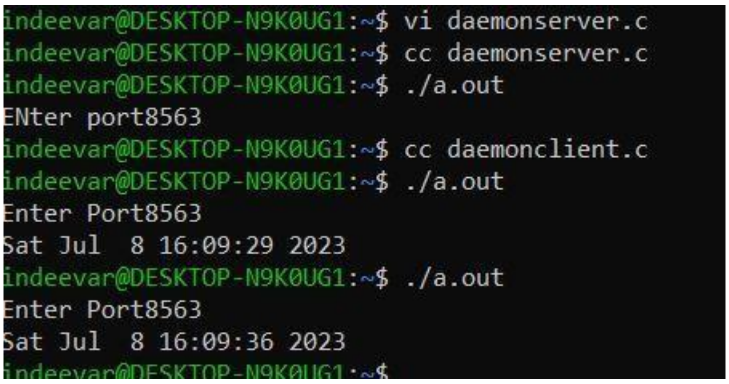
```
        {
                printf("Connect Error");
        }
        n=recv(sock,recv_data,1024,0);
        recv_data[n]='\0';
        printf("%s",recv_data);
        close(sock);
        return 0;
}
```

DaemonServer:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<signal.h>
#include<syslog.h>
#include<time.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define MAXFD 64
int daemon_init(const char *pname,int facility)
{
	int i;
	pid_t pid;
	if((pid=fork())<0)
	{
		return (-1);
	}
	else if(pid)
	{
		_exit(0);
	}
```

```
        if(setsid()<0)
        {
                return -1;
        }
        signal(SIGHUP,SIG_IGN);
        if((pid=fork())<0)
        {
                return (-1);
        }
        else if(pid)
        {
                _exit(0);
        }
        chdir("/");
        for(i=0;i<MAXFD;i++)
        {
                close(i);
        }
        open("/dev/null",444);
        open("/dev/null",666);
        open("/dev/null",O_RDWR);
        openlog(pname,LOG_PID,facility);
        return (0);
}
int main(int argc,char **argv)
{
        int listenfd,connect;
        socklen_t addrlen,len;
        struct sockaddr_in serveraddr,clientaddr;
        char buff[1024];
        time_t ticks;
        int port;
        printf("ENter port");
        scanf("%d",&port);
        daemon_init(argv[0],0);
        listenfd=socket(AF_INET,SOCK_STREAM,0);
        serveraddr.sin_family=AF_INET;
        serveraddr.sin_port=htons(port);
        serveraddr.sin_addr.s_addr=htonl(INADDR_ANY);
        bind(listenfd,(struct sockaddr*)&serveraddr,sizeof(struct sockaddr));
        listen(listenfd,5);
        while(1)
        {
                int sin_size=sizeof(struct sockaddr_in);
                connect=accept(listenfd,(struct sockaddr*)&clientaddr,&sin_size);
                time(&ticks);
                send(connect,ctime(&ticks),strlen(ctime(&ticks)),0);
                close(connect);
        }
        return 0;
}
```

Output:


```
indeevar@DESKTOP-N9K0UG1:~$ vi daemonserver.c
indeevar@DESKTOP-N9K0UG1:~$ cc daemonserver.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
ENter port8563
indeevar@DESKTOP-N9K0UG1:~$ cc daemonclient.c
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Port8563
Sat Jul  8 16:09:29 2023
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
Enter Port8563
Sat Jul  8 16:09:36 2023
indeevar@DESKTOP-N9K0UG1:~$
```

10. Implementation of TCP echo server using threads.

SourceCode:

Threadclient:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
#include<pthread.h>
void str_cli(int);
int sockfd;
void *copyto(void *);
main( )
{
        int port;
        struct sockaddr_in servaddr;
      printf("enter port number:");
      scanf("%d",&port);
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

```
          bzero(&servaddr, sizeof(servaddr));
          servaddr.sin_family = AF_INET;
          servaddr.sin_port = htons(port);
        servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
          connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr));

          str_cli(sockfd);              /* do it all */

          exit(0);
}
void str_cli(int sockfd)
{
        char recv_data[1024];
       pthread_t tid;
        int bytes_received,i;
         for( ;  ; )
         {
         pthread_create(&tid,NULL,copyto,NULL);

         bytes_received=recv(sockfd, recv_data, 1024,0);
         recv_data[bytes_received]='\0';
          printf("received data=%s",recv_data);
        }
//fflush(stdout);
 close(sockfd);
          return;
```

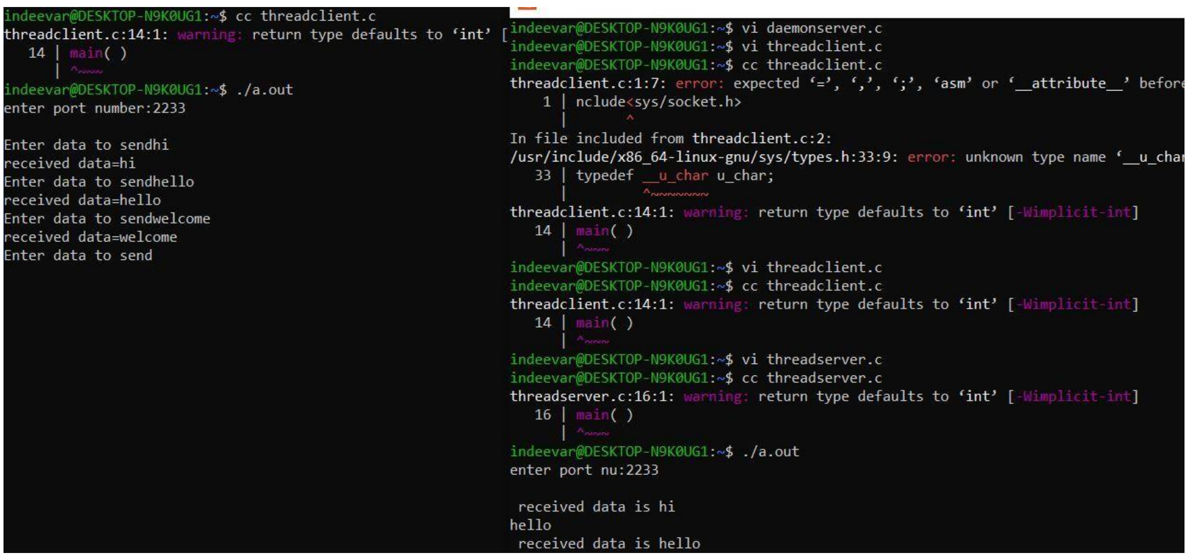
```
}
void *copyto(void *arg)
{
             char  send_data[1024];
              char  recv_data[1024];
              int i,bytes_received;
              //fflush(stdout);

             printf("\nEnter data to send");
            scanf(" %s",send_data);
            send(sockfd,send_data,strlen(send_data),0);
}
```

ThreadServer:

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <time.h>
#include <sys/wait.h>
#include<signal.h>
#include<pthread.h>
void  str_echo(int);
static void *doit(void *);
main( )
{
          int      listenfd, connfd,port,*iptr;
          socklen_t clilen;
        pthread_t tid;
          struct sockaddr_in cliaddr, servaddr;
           printf("enter port nu:");
       scanf("%d",&port);
          listenfd = socket(AF_INET, SOCK_STREAM, 0);
          bzero(&servaddr, sizeof(servaddr));
          servaddr.sin_family      = AF_INET;
          servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
          servaddr.sin_port        = htons(port);

          bind(listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));

          listen(listenfd, 5);
        fflush(stdout);

          for ( ; ; )
        {
                    clilen = sizeof(cliaddr);
                 iptr=malloc(sizeof(int));
               *iptr=accept(listenfd,(struct sockaddr *)&cliaddr,&clilen);
               pthread_create(&tid,NULL,&doit,iptr);
                                            /* parent closes connected socket */
          }
exit(0);
}
static void *doit(void *arg)
{
   int    connfd;

   connfd = *((int *) arg);
   free(arg);
   pthread_detach(pthread_self());
   str_echo(connfd);          /* same function as before */
```

```
    close(connfd);              /* done with connected socket */
    //return (NULL);

}
void str_echo(int sockfd)
{

       char     recv_data[1024],send_data[1024];
       int  i,bytes_received;
while(1)
{
again:
    for( ; ;)
   {
       bytes_received = recv(sockfd, recv_data, sizeof(recv_data),0);
       recv_data[bytes_received]='\0';
     if (bytes_received<0&&errno == EINTR)
          goto again;
      else if (bytes_received<0)
          printf("str_echo: read error");


       printf("\n received data is %s",recv_data);
       //printf("\n enter data to send");
       //scanf("%s",send_data);
       send(sockfd, recv_data, strlen(recv_data),0);
   }
        fflush(stdout);
        return;
}
}
```

Output:



```
indeevar@DESKTOP-N9K0UG1:~$ cc threadclient.c
threadclient.c:14:1: warning: return type defaults to ‘int’ [
   14 | main( )
      | ^~~~
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
enter port number:2233

Enter data to sendhi
received data=hi
Enter data to sendhello
received data=hello
Enter data to sendwelcome
received data=welcome
Enter data to send
```

```
indeevar@DESKTOP-N9K0UG1:~$ vi daemonserver.c
indeevar@DESKTOP-N9K0UG1:~$ vi threadclient.c
indeevar@DESKTOP-N9K0UG1:~$ cc threadclient.c
threadclient.c:1:7: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before
    1 | nclude<sys/socket.h>
      |       ^
In file included from threadclient.c:2:
/usr/include/x86_64-linux-gnu/sys/types.h:33:9: error: unknown type name ‘__u_char
   33 | typedef __u_char u_char;
      |         ^~~~~~~~
threadclient.c:14:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   14 | main( )
      | ^~~~
indeevar@DESKTOP-N9K0UG1:~$ vi threadclient.c
indeevar@DESKTOP-N9K0UG1:~$ cc threadclient.c
threadclient.c:14:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   14 | main( )
      | ^~~~
indeevar@DESKTOP-N9K0UG1:~$ vi threadserver.c
indeevar@DESKTOP-N9K0UG1:~$ cc threadserver.c
threadserver.c:16:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   16 | main( )
      | ^~~~
indeevar@DESKTOP-N9K0UG1:~$ ./a.out
enter port nu:2233

 received data is hi
hello
 received data is hello
```