Exp3

Server:

```
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <string.h>
#include <sys/socket.h>
#include <stdlib.h>
#define PORT 8000

int main (int argc, char const *argv[] )
{
int obj_server, sock, reader;
struct sockaddr_in address;
int opted = 1;
int address_length = sizeof(address);
char buffer[1024] = {0};
char *message= "A Hello Message from server !";

if (( obj_server = socket ( AF_INET, SOCK_STREAM, 0)) == 0)
{
perror( "Opening of Socket Failed !");
exit(EXIT_FAILURE);
}

if (setsockopt(obj_server, SOL_SOCKET, SO_REUSEADDR,&opted, sizeof ( opted )))
{
   perror("Can't set the socket" );
   exit (EXIT_FAILURE );
}

address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( PORT );

if (bind(obj_server, ( struct sockaddr * )&address, sizeof(address))<0)
{
   perror ( "Binding of socket failed !" );
   exit(EXIT_FAILURE);
}

if (listen ( obj_server, 3) < 0)
{
   perror ( "Can't listen from the server !");
   exit(EXIT_FAILURE);
}

if ((sock = accept(obj_server, (struct sockaddr *)&address, (socklen_t*)&address_length)) < 0)
```

```
{
   perror("Accept");
   exit(EXIT_FAILURE);
}

reader = read(sock, buffer, 1024);
printf("%s\n", buffer);
send(sock , message, strlen(message) , 0 );
printf("Server : Message has been sent ! \n");
return 0;
}
```

Client:

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8000

int main (int argc, char const *argv[] )
{
int obj_socket = 0, reader;
struct sockaddr_in serv_addr;

char *message = "A Hello message from Client !";
char buffer[1024] = {0};

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

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

if(inet_pton( AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)
{
printf ( "\nInvalid address ! This IP Address is not supported !\n" );
return -1;
}
if (connect( obj_socket, (struct sockaddr *)&serv_addr, sizeof(serv_addr )) < 0)
{
printf ( "Connection Failed : Can't establish a connection over this socket !" );
return -1;
}
send( obj_socket, message, strlen(message) , 0 );
printf ( "\nClient : Message has been sent !\n" );
```

```
reader = read ( obj_socket, buffer, 1024 );
printf ( "%s\n",buffer );
return 0;
}
```

Output:

```
Stop                Command: exp\ -\ 3/60\ tcp_server1.c

Server listening..
server accept the client...
From client: Hellow
        To client : hii
```

```
Stop                Command: exp\ -\ 3/60\ tcp_client1.c

Socket successfully created..
connected to the server..
Enter the string : Hellow
From Server : hii
Enter the string :
```

Exp4

Server:

```
 // Client side implementation of UDP client-server model
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT  8080
#define MAXLINE 1024

// Driver code
int main() {
 int sockfd;
 char buffer[MAXLINE];
 char *hello = "Hello from client";
 struct sockaddr_in  servaddr;

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

 memset(&servaddr, 0, sizeof(servaddr));

 // Filling server information
 servaddr.sin_family = AF_INET;
 servaddr.sin_port = htons(PORT);
 servaddr.sin_addr.s_addr = INADDR_ANY;

 int n, len;

 sendto(sockfd, (const char *)hello, strlen(hello),
  MSG_CONFIRM, (const struct sockaddr *) &servaddr,
   sizeof(servaddr));
 printf("Hello message sent.\n");

 n = recvfrom(sockfd, (char *)buffer, MAXLINE,
    MSG_WAITALL, (struct sockaddr *) &servaddr,
    &len);
 buffer[n] = '\0';
 printf("Server : %s\n", buffer);

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

Client:

```
// Server side implementation of UDP client-server model
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT  8080
#define MAXLINE 1024

// Driver code
int main() {
 int sockfd;
 char buffer[MAXLINE];
 char *hello = "Hello from server";
 struct sockaddr_in servaddr, cliaddr;

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

 memset(&servaddr, 0, sizeof(servaddr));
 memset(&cliaddr, 0, sizeof(cliaddr));

 // Filling server information
 servaddr.sin_family = AF_INET; // IPv4
 servaddr.sin_addr.s_addr = INADDR_ANY;
 servaddr.sin_port = htons(PORT);

 // Bind the socket with the server address
 if ( bind(sockfd, (const struct sockaddr *)&servaddr,
   sizeof(servaddr)) < 0 )
 {
  perror("bind failed");
  exit(EXIT_FAILURE);
 }

 int len, n;

 len = sizeof(cliaddr); //len is value/result

 n = recvfrom(sockfd, (char *)buffer, MAXLINE,
    MSG_WAITALL, ( struct sockaddr *) &cliaddr,
    &len);
 buffer[n] = '\0';
```

```
 printf("Client : %s\n", buffer);
 sendto(sockfd, (const char *)hello, strlen(hello),
  MSG_CONFIRM, (const struct sockaddr *) &cliaddr,
   len);
 printf("Hello message sent.\n");

 return 0;
}
```

Output:

Exp5

Server:

```
#include<netinet/in.h>
#include<sys/socket.h>
#include<stdio.h>
#include<string.h>
#include<time.h>
int main( )
{
struct sockaddr_in sa;
struct sockaddr_in cli;
int sockfd,conntfd;
int len,ch;
char str[100];
time_t tick;
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0)
{
printf("error in socket\n");
exit(0);
}
else printf("Socket opened");
bzero(&sa,sizeof(sa));
sa.sin_port=htons(5600);
sa.sin_addr.s_addr=htonl(0);
if(bind(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0)
{
printf("Error in binding\n");
}
else
printf("Binded Successfully");
listen(sockfd,50);
for(;;)
{
len=sizeof(ch);
conntfd=accept(sockfd,(struct sockaddr*)&cli,&len);
printf("Accepted");
tick=time(NULL);
snprintf(str,sizeof(str),"%s",ctime(&tick));
printf("%s",str);write(conntfd,str,100);
}
}
```

Client:

```
#include"netinet/in.h"
#include"sys/socket.h"
#include"stdio.h"
main()
{
struct sockaddr_in sa,cli;
int n,sockfd;
```

```
int len;char buff[100];
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0)
{ printf("\nError in Socket");
exit(0);
}
else printf("\nSocket is Opened");
bzero(&sa,sizeof(sa));
sa.sin_family=AF_INET;
sa.sin_port=htons(5600);
if(connect(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0)
{
printf("\nError in connection failed");
exit(0);
}
else
printf("\nconnected successfully");
if(n=read(sockfd,buff,sizeof(buff))<0)
{
printf("\nError in Reading");
exit(0);
}
else
{printf("\nMessage Read %s",buff);
}}
```

```
Run                Command: Exp5/60dtserver.c

Server is running...

Process exited with code: 141
```

```
PMahalakshmi:~/environment/Exp5 $ ./a.out 127.0.0.1
Day time of server is: Fri Oct 28 06:03:00 2022
```

Exp6:

Server:

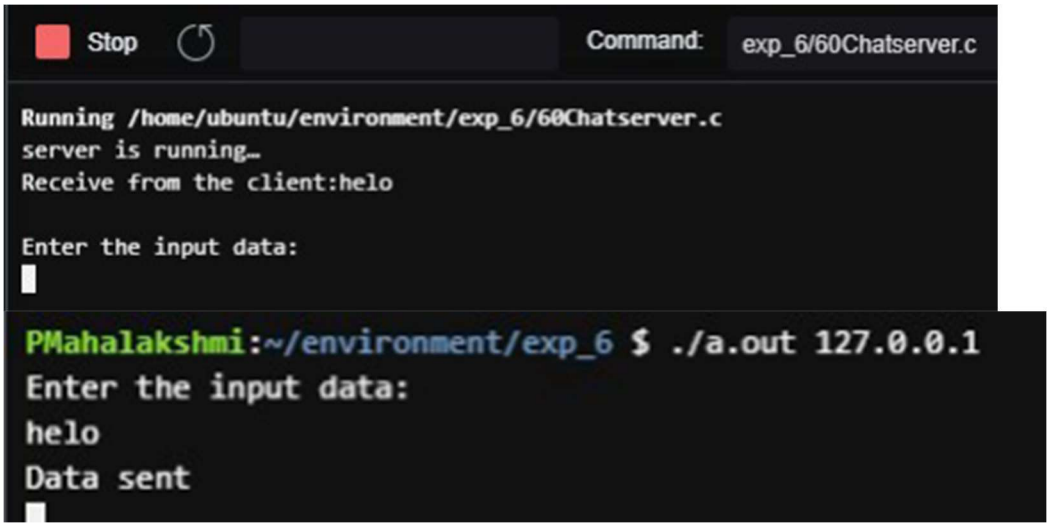
```
#include<sys/types.h>
#include<stdio.h>
#include<netdb.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<string.h>
#include<unistd.h>
#include<netinet/in.h>
int main(int argc,char *argv[])
{
int n,sd,ad;
struct sockaddr_in servaddr,cliaddr;
socklen_t clilen,servlen;
char buff[10000],buff1[10000];
bzero(&servaddr,sizeof(servaddr));

        /*Socket address structure*/
          servaddr.sin_family=AF_INET;
          servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
          servaddr.sin_port=htons(5000);

        /*TCP socket is created, an Internet socket address structure is filled with wildcard
address & server's well known port*/
          sd=socket(AF_INET,SOCK_STREAM,0);

          /*Bind function assigns a local protocol address to the socket*/
          bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));

          /*Listen function specifies the maximum number of connections that kernel
          should queue for this socket*/
          listen(sd,5);
          printf("%s\n","server is running…");

          /*The server to return the next completed connection from the front of the
        completed connection Queue calls it*/
          ad=accept(sd,(struct sockaddr*)&cliaddr,&clilen);
          while(1)
          {
                    bzero(&buff,sizeof(buff));

                    /*Receiving the request from client*/
                    recv(ad,buff,sizeof(buff),0);

                    printf("Receive from the client:%s\n",buff);
                    n=1;
                    while(n==1)
                    {
                              bzero(&buff1,sizeof(buff1));
                              printf("%s\n","Enter the input data:");
```

```
                        /*Read the message from client*/
                        fgets(buff1,10000,stdin);

                        /*Sends the message to client*/
                        send(ad,buff1,strlen(buff1)+1,0);
                        printf("%s\n","Data sent");
                        n=n+1;
                }
        }
        return 0;
}
```

Client:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>
#include<unistd.h>
#include<stdio.h>
#include<netdb.h>
int main(int argc,char *argv[])
{
        int n,sd,cd;
        struct sockaddr_in servaddr,cliaddr;
        socklen_t servlen,clilen;
        char buff[10000],buff1[10000];
        bzero(&servaddr,sizeof(servaddr));

        /*Socket address structure*/
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=inet_addr(argv[1]);
        servaddr.sin_port=htons(5000);

        /*Creating a socket, assigning IP address and port number for that socket*/
        sd=socket(AF_INET,SOCK_STREAM,0);

        /*Connect establishes connection with the server using server IP address*/
        cd=connect(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
        while(1)
        {
                bzero(&buff,sizeof(buff));
                printf("%s\n","Enter the input data:");

                /*This function is used to read from server*/
                fgets(buff,10000,stdin);
```

```
		/*Send the message to server*/
		send(sd,buff,strlen(buff)+1,0);
		printf("%s\n","Data sent");
		n=1;
		while(n==1)
		{
			bzero(&buff1,sizeof(buff1));

			/*Receive the message from server*/
			recv(sd,buff1,sizeof(buff1),0);
			printf("Received from the server:%s\n",buff1);
			n=n+1;
		}
	}
	return 0;
}
```



Exp7
Server:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>
int main(int argc,char *argv[])
{
int ad,sd;
struct sockaddr_in servaddr,cliaddr;
socklen_t servlen,clilen;
char buff[1000],buff1[1000];
pid_t cpid;
bzero(&servaddr,sizeof(servaddr));

        /*Socket address structure*/
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(9652);

            /*TCP socket is created, an Internet socket address structure is filled with            wildcard
address & server's well known port*/
sd=socket(AF_INET,SOCK_STREAM,0);

            /*Bind function assigns a local protocol address to the socket*/
bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));

            /*Listen function specifies the maximum number of connections that kernel        should
queue for this socket*/
listen(sd,5);
printf("%s\n","Server is running.....");

          /*The server to return the next completed connection from the front of the
           completed connection Queue calls it*/
ad=accept(sd,(struct sockaddr*)&cliaddr,&clilen);

            /*Fork system call is used to create a new process*/
 cpid=fork();



if(cpid==0)
{
                    while(1)
                    {
                              bzero(&buff,sizeof(buff));
```

```
                        /*Receiving the request from client*/
                        recv(ad,buff,sizeof(buff),0);
                printf("Received  message from the client:%s\n",buff);
        }
}
else
{
        while(1)
        {
                bzero(&buff1,sizeof(buff1));
                printf("%s\n","Enter the input data:");

                /*Read the message from client*/
                fgets(buff1,10000,stdin);

                /*Sends the message to client*/
                send(ad,buff1,strlen(buff1)+1,0);
                printf("%s\n","Data sent…");
        }
}
return 0;
}
```

Client:

```
#include<sys/socket.h>
#include<sys/types.h>
#include<stdio.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<netdb.h>
#include<netinet/in.h>
#include<string.h>
int main(int argc,char *argv[])
{
        int sd,cd;
        struct sockaddr_in servaddr,cliaddr;
        socklen_t servlen,clilen;
        char buff[1000],buff1[1000];
        pid_t cpid;
        bzero(&servaddr,sizeof(servaddr));
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=inet_addr(argv[1]);
        servaddr.sin_port=htons(9652);

        /*Creating a socket, assigning IP address and port number for that socket*/
        sd=socket(AF_INET,SOCK_STREAM,0);

        /*Connect establishes connection with the server using server IP address*/
        cd=connect(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));

        /*Fork is used to create a new process*/
```

```
        cpid=fork();
        if(cpid==0)
        {
                while(1)
                {
                        bzero(&buff,sizeof(buff));
                        printf("%s\n","Enter the input data:");

                        /*This function is used to read from server*/
                        fgets(buff,10000,stdin);

                         /*Send the message to server*/
                        send(sd,buff,strlen(buff)+1,0);
                        printf("%s\n","Data sent…");
                }
        }

        else
        {
                while(1)
                {
                        bzero(&buff1,sizeof(buff1));

                        /*Receive the message from server*/
                        recv(sd,buff1,sizeof(buff1),0);
                        printf("Received message from the server:%s\n",buff1);
                }
        }
        return 0;
}
```

Output:

```
Stop        Command: server7.c

Running /home/ubuntu/environment/server7.c
/home/ubuntu/environment/server7.c: In function 'main':
/home/ubuntu/environment/server7.c:16:1: warning: implicit declaration of function 'bzero' [-Wimplicit-function-declaration]
 bzero(&servaddr,sizeof(servaddr));
 ^~~~~
Server is running.......
Enter the input data:
Received  message from the client:hii
```

```
RA2011030010104:~/environment/$ cc client7.c
RA2011030010104:~/environment/$ ./a.out 127.0.0.1
Enter the input data:
hii
Data sent…
Enter the input data:
```

Exp8

Server:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#define SIZE 1024

void write_file(int sockfd){
  int n;
  FILE *fp;
  char *filename = "recv1.txt";
  char buffer[SIZE];

  fp = fopen(filename, "w");
  while (1) {
    n = recv(sockfd, buffer, SIZE, 0);
    if (n <= 0){
      break;
      return;
    }
    fprintf(fp, "%s", buffer);
    bzero(buffer, SIZE);
  }
  return;
}

int main(){
  char *ip = "127.0.0.1";
  int port = 1500;
  int e;

  int sockfd, new_sock;
  struct sockaddr_in server_addr, new_addr;
  socklen_t addr_size;
  char buffer[SIZE];

  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if(sockfd < 0) {
    perror("[-]Error in socket");
    exit(1);
  }
  printf("[+]Server socket created successfully.\n");

  server_addr.sin_family = AF_INET;
  server_addr.sin_port = port;
  server_addr.sin_addr.s_addr = inet_addr(ip);

  e = bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
  if(e < 0) {
    perror("[-]Error in bind");
```

```
    exit(1);
  }
  printf("[+]Binding successfull.\n");

  if(listen(sockfd, 10) == 0){
 printf("[+]Listening....\n");
 }else{
 perror("[-]Error in listening");
    exit(1);
 }

  addr_size = sizeof(new_addr);
  new_sock = accept(sockfd, (struct sockaddr*)&new_addr, &addr_size);
  write_file(new_sock);
  printf("[+]Data written in the file successfully.\n");

  return 0;
}
```

Client:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#define SIZE 1024

void send_file(FILE *fp, int sockfd){
  int n;
  char data[SIZE] = {0};

  while(fgets(data, SIZE, fp) != NULL) {
    if (send(sockfd, data, sizeof(data), 0) == -1) {
      perror("[-]Error in sending file.");
      exit(1);
    }
    bzero(data, SIZE);
  }
}

int main(){
  char *ip = "127.0.0.1";
  int port = 1500;
  int e;

  int sockfd;
  struct sockaddr_in server_addr;
  FILE *fp;
  char *filename = "it.txt";

  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if(sockfd < 0) {
```

```
  perror("[-]Error in socket");
  exit(1);
 }
 printf("[+]Server socket created successfully.\n");

 server_addr.sin_family = AF_INET;
 server_addr.sin_port = port;
 server_addr.sin_addr.s_addr = inet_addr(ip);

 e = connect(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
 if(e == -1) {
  perror("[-]Error in socket");
  exit(1);
 }
printf("[+]Connected to Server.\n");

 fp = fopen(filename, "r");
 if (fp == NULL) {
  perror("[-]Error in reading file.");
  exit(1);
 }

 send_file(fp, sockfd);
 printf("[+]File data sent successfully.\n");

 printf("[+]Closing the connection.\n");
 close(sockfd);

 return 0;
}
```

Output:

```
Running /home/ubuntu/environment/server8.c
/home/ubuntu/environment/server8.c: In function 'write_file':
/home/ubuntu/environment/server8.c:21:5: warning: implicit declaration of function 'bzero' [-Wimplicit-function-declaration]
     bzero(buffer, SIZE);
     ^~~~~
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening....
[+]Data written in the file successfully.
```

```
RA2011030010104:~/environment$ cc client8.c
RA2011030010104:~/environment$ ./a.out 127.0.0.1
[+]Server socket created successfully.
[+]Connected to Server.
[-]Error in reading file.
```

Exp9

Server:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<netdb.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/stat.h>
#include<arpa/inet.h>
#include<unistd.h>
int main(int argc,char* argv[]){
int sd,size;
char buff[1024],file[10000];
struct sockaddr_in cliaddr,servaddr;
FILE *fp;
struct stat x;
socklen_t clilen;
clilen=sizeof(cliaddr);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(9976);
sd=socket(AF_INET,SOCK_DGRAM,0);
if(sd<0)
{
printf("Socket CReation Error");
}
bind(sd,(struct sockaddr *)&servaddr,sizeof(servaddr));
while(1)
{
bzero(buff,sizeof(buff));
recvfrom(sd,buff,sizeof(buff),0,(struct sockaddr *)&cliaddr,&clilen);

strcat(buff,">file1");
system(buff);
fp=fopen("file1","r");
stat("file1",&x);
size=x.st_size;
fread(file,size,1,fp);

sendto(sd,file,sizeof(file),0,(struct sockaddr *)&cliaddr,sizeof(cliaddr));
printf("Data Sent to UDPCLIENT %s",buff);
}
close(sd);
return 0; }
```

Client:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<netdb.h>
#include<netinet/in.h>
#include<string.h>
#include<arpa/inet.h>
#include<sys/stat.h>
int main(int argc,char* argv[]){
int sd;
char buff[1024],file[10000];
struct sockaddr_in cliaddr,servaddr;
struct hostent *h;
socklen_t servlen;
servlen=sizeof(servaddr);
h=gethostbyname(argv[1]);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=h->h_addrtype;
memcpy((char *)&servaddr.sin_addr,h->h_addr_list[0],h->h_length);
servaddr.sin_port=htons(9976);
sd=socket(AF_INET,SOCK_DGRAM,0);
if(sd<0)
{
printf("Socket CReation Error");
}
bind(sd,(struct sockaddr *)&servaddr,sizeof(servaddr));
while(1)
{
printf("\nEnter the command to be executed");
fgets(buff,1024,stdin);
sendto(sd,buff,strlen(buff)+1,0,(struct sockaddr *)&servaddr,sizeof(servaddr));
printf("\nData Sent");
recvfrom(sd,file,strlen(file)+1,0,(struct sockaddr *)&servaddr,&servlen);
printf("Recieved From UDPSERVER %s",file);
}
return 0;
}
```

Output:

```
Running /home/ubuntu/environment/server9.c
/home/ubuntu/environment/server9.c: In function 'main':
/home/ubuntu/environment/server9.c:19:9: warning: implicit declaration of function 'bzero' [-Wimplicit-function-declaration]
         bzero(&servaddr,sizeof(servaddr));
         ^~~~~
/home/ubuntu/environment/server9.c:34:3: warning: implicit declaration of function 'system'; did you mean 'listen'? [-Wimplicit-function-declaration]
   system(buff);
   ^~~~~~
   listen
ARP10.c    client.c    client5.c    client6.py  client8.c.o  file1      server.c.o   server5.c    server6.c.o  server7.c.o  server9.c
ARP10.c.o  client.c.o  client6.c    client7.c   client9.c    recv1.txt  server4.c    server5.c.o  server6.py   server8.c    server9.c.o
a.out      client4.c   client6.c.o  client8.c   client9.c.o  server.c   server4.c.o  server6.c    server7.c    server8.c.o
Data Sent to UDPCLIENT ls
```

```
RA2011030010104:~/environment$ cc client9.c
RA2011030010104:~/environment$ ./a.out 127.0.0.1

Enter the command to be executedls

Data Sent
Recieved From UDPSERVER
Enter the command to be executed
```

Exp10

Server:

```
#include<sys/types.h>
#include<stdio.h>
#include<netdb.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<string.h>
#include<unistd.h>
#include<netinet/in.h>
int main(int argc,char *argv[])
{
int n,sd,ad;
struct sockaddr_in servaddr,cliaddr;
socklen_t clilen,servlen;
char buff[10000],buff1[10000];
bzero(&servaddr,sizeof(servaddr));

        /*Socket address structure*/
          servaddr.sin_family=AF_INET;
          servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
          servaddr.sin_port=htons(5000);

        /*TCP socket is created, an Internet socket address structure is filled with wildcard address &
server's well known port*/
          sd=socket(AF_INET,SOCK_STREAM,0);

          /*Bind function assigns a local protocol address to the socket*/
          bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));

          /*Listen function specifies the maximum number of connections that kernel
          should queue for this socket*/
          listen(sd,5);
          printf("%s\n","server is running…");

          /*The server to return the next completed connection from the front of the
        completed connection Queue calls it*/
          ad=accept(sd,(struct sockaddr*)&cliaddr,&clilen);
          while(1)
          {
                    bzero(&buff,sizeof(buff));

                    /*Receiving the request from client*/
                    recv(ad,buff,sizeof(buff),0);

                    printf("Receive from the client:%s\n",buff);
                    n=1;
                    while(n==1)
                    {
                              bzero(&buff1,sizeof(buff1));
                              printf("%s\n","Enter the input data:");
```

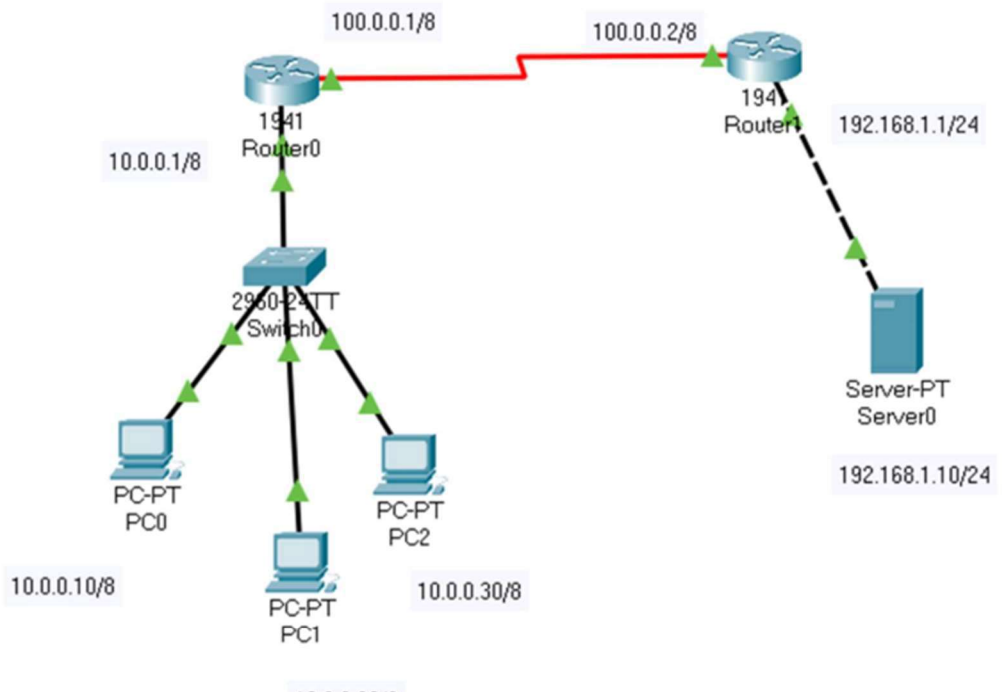
```
                        /*Read the message from client*/
                        fgets(buff1,10000,stdin);

                        /*Sends the message to client*/
                        send(ad,buff1,strlen(buff1)+1,0);
                        printf("%s\n","Data sent");
                        n=n+1;
                }
        }
        return 0;
}
```

Client:

```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<unistd.h>
#include<netinet/in.h>
#include<netdb.h>
#include<arpa/inet.h>
int main(int argc,char * argv[])
{
int cd,sd,ad;
char buff[1024];
struct sockaddr_in cliaddr,servaddr;
struct hostent *h;
        /*This function looks up a hostname and it returns a pointer to a hostent
      structure that contains all the IPV4 address*/
h=gethostbyname(argv[1]);
bzero(&servaddr,sizeof(servaddr));

/*Socket address structure*/
servaddr.sin_family=AF_INET;
memcpy((char *)&servaddr.sin_addr.s_addr,h->h_addr_list[0],h->h_length);
servaddr.sin_port = htons(1999);

      /*Creating a socket, assigning IP address and port number for that socket*/
sd = socket(AF_INET,SOCK_STREAM,0);

       /*Connect establishes connection with the server using server IP address*/
cd=connect(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
while(1)
{
printf("Enter the message: \n");

 /*Reads the message from standard input*/
 fgets(buff,100,stdin);

          /*Send function is used on client side to send data given by user on client
```

```
            side to the server*/
send(sd,buff,sizeof(buff)+1,0);
printf("\n Data Sent ");
//recv(sd,buff,strlen(buff)+1,0);
printf("%s",buff);
}}
```

```
RA2011030010104:~/environment$ cc ARP10.c
ARP10.c: In function ‘main’:
ARP10.c:20:1: warning: implicit declaration of function ‘exit’ [-Wimplicit-function-declaration]
 exit(0);
 ^~~~
ARP10.c:20:1: warning: incompatible implicit declaration of built-in function ‘exit’
ARP10.c:20:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
ARP10.c:28:1: warning: incompatible implicit declaration of built-in function ‘exit’
 exit(0);
 ^~~~
ARP10.c:28:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
ARP10.c:32:8: warning: too many arguments for format [-Wformat-extra-args]
 printf("%X:%X:%X:%X:%X:%X\n",*ptr,*(ptr+1),*(ptr+2),*(ptr+3),*(ptr+4),*(ptr+5),*(ptr+5));
        ^~~~~~~~~~~~~~~~~~~~~~~
RA2011030010104:~/environment$ ./a.out 127.0.0.1
MAC Address For '127.0.0.1' : 0:0:0:0:0:0
RA2011030010104:~/environment$
```

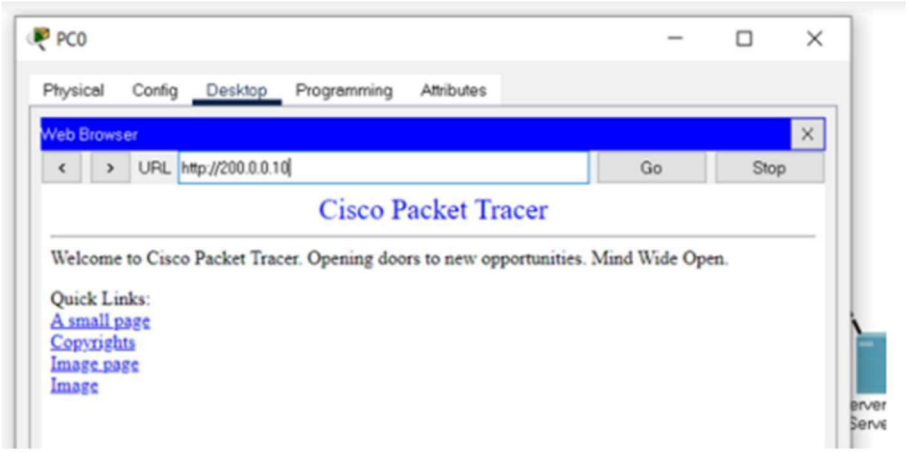


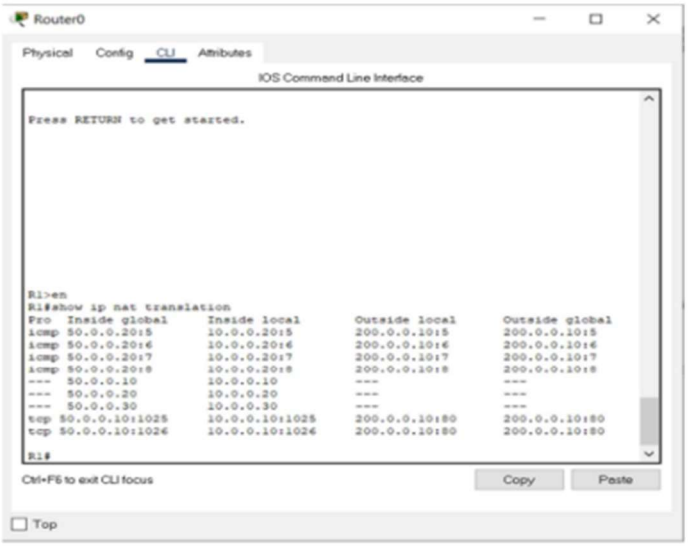
Physical Config Desktop Programming Attributes

Command Prompt

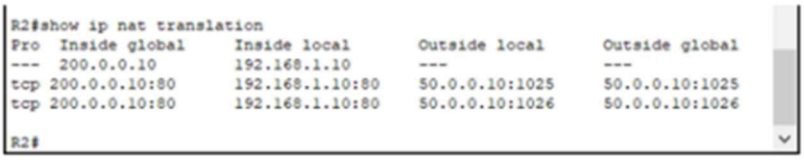
```
Reply from 10.0.0.1: Destination host unreachable.

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>ipconfig

FastEthernet0 Connection:(default port)

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.........: FE80::260:47FF:FE93:623B
   IPv6 Address....................: ::
   IPv4 Address....................: 10.0.0.10
   Subnet Mask.....................: 255.0.0.0
   Default Gateway.................: ::
                                     10.0.0.1

Bluetooth Connection:

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.........: ::
   IPv6 Address....................: ::
   IPv4 Address....................: 0.0.0.0
   Subnet Mask.....................: 0.0.0.0
   Default Gateway.................: ::
                                     0.0.0.0

C:\>ping 200.0.0.10

Pinging 200.0.0.10 with 32 bytes of data:

Reply from 200.0.0.10: bytes=32 time=10ms TTL=126
Reply from 200.0.0.10: bytes=32 time=1ms TTL=126
Reply from 200.0.0.10: bytes=32 time=2ms TTL=126
Reply from 200.0.0.10: bytes=32 time=8ms TTL=126

Ping statistics for 200.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 10ms, Average = 5ms

C:\>ping 192.168.1.10

Pinging 192.168.1.10 with 32 bytes of data:

Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
```

Top

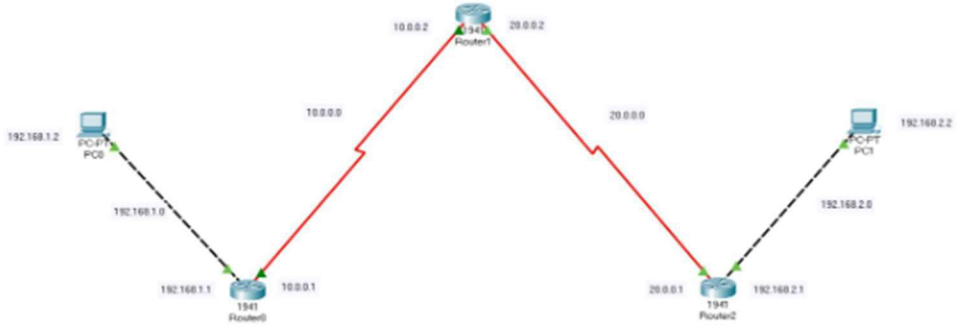




For router1:



```
R2#show ip nat translation
Pro  Inside global      Inside local        Outside local       Outside global
---  200.0.0.10         192.168.1.10        ---                 ---
tcp 200.0.0.10:80       192.168.1.10:80     50.0.0.10:1025      50.0.0.10:1025
tcp 200.0.0.10:80       192.168.1.10:80     50.0.0.10:1026      50.0.0.10:1026

R2#
```



PC0

Physical Config Desktop Programming Attributes

Command Prompt

```
Reply from 192.168.2.2: bytes=32 time=22ms TTL=125
Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=25ms TTL=125

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 25ms, Average = 12ms

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>cls
Invalid Command.

C:\>tracert 192.168.2.2

Tracing route to 192.168.2.2 over a maximum of 30 hops:

  1   0 ms      0 ms      0 ms      192.168.1.1
  2   8 ms      10 ms     12 ms     172.18.1.2
  3   5 ms      2 ms      10 ms     192.168.2.2

Trace complete.
```

PC0

Physical Config Desktop Programming Attributes

Command Prompt

```
Packet Tracer PC Command Line 1.0
C:\>ping 192.268.2.2
Ping request could not find host 192.268.2.2. Please check the name and try
again.
C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=22ms TTL=125
Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=25ms TTL=125

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 25ms, Average = 12ms

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
```

Router1

Physical Config CLI Attributes

IOS Command Line Interface

```
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#show interfaces tunnel 200
Tunnel200 is up, line protocol is up (connected)
  Hardware is Tunnel
  Internet address is 172.18.1.1/16
  MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 10.0.0.1 (Serial0/1/0), destination 20.0.0.2
  Tunnel protocol/transport GRE/IP
    Key disabled, sequencing disabled
    Checksumming of packets disabled
  Tunnel TTL 255
  Fast tunneling enabled
  Tunnel transport MTU 1476 bytes
  Tunnel transmit bandwidth 8000 (kbps)
  Tunnel receive bandwidth 8000 (kbps)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1
  Queueing strategy: fifo
  Output queue: 0/0 (size/max)
  5 minute input rate 8 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     5 packets input, 640 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
```

Ctrl+F6 to exit CLI focus Copy Paste

Router3

Physical Config CLI Attributes

IOS Command Line Interface

```
%SYS-5-CONFIG_I: Configured from console by console

Router#show interfaces Tunnel 200
%Invalid interface type and number

Router#show interfaces Tunnel 400
Tunnel400 is up, line protocol is up (connected)
  Hardware is Tunnel
  Internet address is 172.18.1.2/16
  MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 20.0.0.2 (Serial0/1/0), destination 10.0.0.1
  Tunnel protocol/transport GRE/IP
    Key disabled, sequencing disabled
    Checksumming of packets disabled
  Tunnel TTL 255
  Fast tunneling enabled
  Tunnel transport MTU 1476 bytes
  Tunnel transmit bandwidth 8000 (kbps)
  Tunnel receive bandwidth 8000 (kbps)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1
  Queueing strategy: fifo
  Output queue: 0/0 (size/max)
  5 minute input rate 8 bits/sec, 0 packets/sec
--More--
```

Ctrl+F6 to exit CLI focus Copy Paste

PC0

Physical Config Desktop Programming Attributes

Command Prompt

```
Packet Tracer PC Command Line 1.0
C:\>ping 192.268.2.2
Ping request could not find host 192.268.2.2. Please check the name and try
again.
C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=22ms TTL=125
Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=25ms TTL=125

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 25ms, Average = 12ms

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
```

PC0

Physical Config Desktop Programming Attributes

Command Prompt

```
Reply from 192.168.2.2: bytes=32 time=22ms TTL=125
Reply from 192.168.2.2: bytes=32 time=2ms TTL=125
Reply from 192.168.2.2: bytes=32 time=25ms TTL=125

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 25ms, Average = 12ms

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>cls
Invalid Command.

C:\>tracert 192.168.2.2

Tracing route to 192.168.2.2 over a maximum of 30 hops:

  1   0 ms      0 ms      0 ms      192.168.1.1
  2   8 ms      10 ms     12 ms     172.18.1.2
  3   5 ms      2 ms      10 ms     192.168.2.2

Trace complete.
```


```
Router#ping 192.168.1.6

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 198.168.1.6, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 26/28/33 ms
```

Now we go to Router1 and test the network by pinging the Router0 interface.

```
Router#ping 192.168.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 25/28/32 ms
```


Fa0
Laptop-PT
Laptop0
10.0.0.2
Fa0/0
2811
Router B
192.168.1.2/30
Fa0/0
Router A
192.168.1.3/30
Fa0 aptop-PT
Laptop1
20.0.0.2


Router1
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#
Router(config-if)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#
Router#
Router#
Router#
Router#
Router#ping 192.168.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 5/7/9 ms
Router#
Router#
Router#
Router#
Router#
Router#
Router#
Ctrl+F6 to exit CLI focus
Copy
Paste