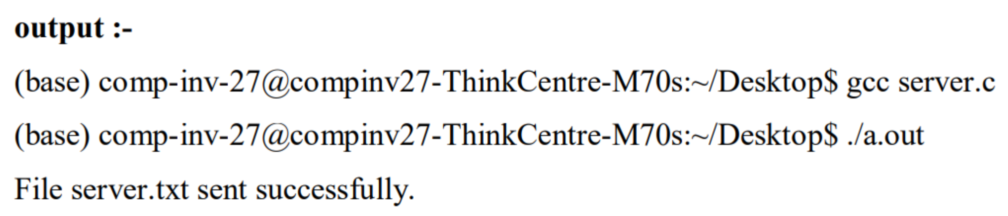
Experiment No. 09
Write a program using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines.

Program for UDP socket:
//add.txt
Hi this is User

//ft_client.c

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

#define ERROR -1
#define BUFFER 1024

int main(int argc, char **argv)
 {
  int sock; // client socket descriptor

  struct sockaddr_in remote_server;

  char send_data[BUFFER]= "add.txt"; // send buffer: hold information to send to server char
  recv_data[BUFFER]; // receive buffer: hold information to received from server


  int sockaddr_len = sizeof(struct sockaddr_in); // socket address length
  int data_len; // store data length of send_data or recv_data length

  if((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
   {
  perror("socket error. ");
  exit(-1);
   }

  remote_server.sin_family = AF_INET; //IPv4
  remote_server.sin_port = htons(atoi(argv[2])); //htons: host to network short byte order, atoi:  convert a string to an
integer
  remote_server.sin_addr.s_addr = inet_addr(argv[1]); //inet_addr: function converts the  Internet host address cp
IPv4 numbers-and-dots notation into binary data in network  byte order.
  bzero(&remote_server.sin_zero , 8);// function sets the first 8 bytes of the area starting
at  &remote_server.sin_zero to zero
  sendto(sock,send_data,sizeof(send_data),0,(struct sockaddr
*)&remote_server,sockaddr_len);

  data_len = recvfrom(sock,recv_data,BUFFER, 0,(struct sockaddr
*)&remote_server,&sockaddr_len);
  recv_data[data_len] = '\0';
printf("data received = %s \n", recv_data );

 FILE *fp;
 fp=fopen("add1.txt","w");
 fprintf(fp,"%s",recv_data);
 fclose(fp);
 close(sock);
 return 0;
 }
```

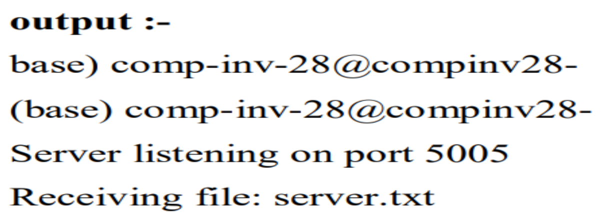


output :-

```
(base) comp-inv-27@compinv27-ThinkCentre-M70s:~/Desktop$ gcc server.c
(base) comp-inv-27@compinv27-ThinkCentre-M70s:~/Desktop$ ./a.out
File server.txt sent successfully.
```

//ft_server.c

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

#define ERROR -1
#define BUFFER 1024
#define MAX_CLIENTS 2

int main(int argc, char **argv)
 {
 int sock, cli; // sock: server socket descriptor, cli: client socket descriptor struct
 sockaddr_in server, client;

 char send_data[BUFFER]= "Hello from server mayur"; // send buffer: hold information to  send to client
 char recv_data[BUFFER]; // receive buffer: hold information to received from client

 int sockaddr_len = sizeof(struct sockaddr_in); // socket address length int data_len; //
 store data length of send_data or recv_data length

 if((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
  {
 perror("socket error. ");
 exit(-1);
  }

 server.sin_family = AF_INET;
 server.sin_port = htons(atoi(argv[1]));
 server.sin_addr.s_addr = INADDR_ANY;
 bzero(&server.sin_zero , 8);

 if((bind(sock, (struct sockaddr *)&server, sockaddr_len)) == -1)
  {
 perror("bind error");
 exit(-1);
  }

// recvfrom(listenfd,buff,buffsize,0, (struct sockaddr *) &clientaddr, &sin_size) !=  buffsize)

                data_len = recvfrom(sock,recv_data,BUFFER, 0,(struct sockaddr *)&client,&sockaddr_len);

 printf("New client connected to port: %d and IP %s \n",
ntohs(client.sin_port),  inet_ntoa(client.sin_addr));

 recv_data[data_len] = '\0';
 FILE *f;
 f=fopen(recv_data,"r");
 fgets(send_data,BUFFER,f);
 fclose(f);

 sendto(sock,send_data,strlen(send_data),0,(struct sockaddr *)&client,sockaddr_len);

 printf("data received = %s \n", recv_data );
 close(sock);
 return 0;
}
```



```
output :-
base) comp-inv-28@compinv28-
(base) comp-inv-28@compinv28-
Server listening on port 5005
Receiving file: server.txt
```