**//add.txt**

High this is Dnyaneshwar

**//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;

}

**//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;

}