Client:

```
RA1911026010114:~/environment/RA1911026010114/CN LAB 8/Client (master) $ cc client.c
RA1911026010114:~/environment/RA1911026010114/CN LAB 8/Client (master) $ ./a.out
File name : send.txt
File Output :
We are from SRM
File name :
```

Result :

Thus the FTP client-server communication is established and data is transferred between the client and server machines.

Ex.No:9	REMOTE COMMAND EXECUTION USING UDP
Date:	

Aim:

There are two hosts, Client and Server. The Client sends a command to the Server, which executes the command and sends the result back to the Client.

TECHNICAL OBJECTIVE:

Remote Command execution is implemented through this program using which Client is able to execute commands at the Server. Here, the Client sends the command to the Server for remote execution. The Server executes the command and the send result of the execution back to the Client.

METHODOLOGY:

Server:

- Include the necessary header files.
- Create a socket using socket function with family AF_INET, type as SOCK_DGRAM.
- Initialize server address to 0 using the bzero function.
- Assign the sin _family to AF_INET, sin_addr to INADDR_ANY, sin _port to dynamicallyassigned port number.
- Bind the local host using the bind() system call.
- Within an infinite loop, receive the command to be executed from the client.
- Append text "> temp.txt" to the command.
- Execute the command using the "system()" system call.
- Send the result of execution to the Client using a file buffer.

Client:

- Include the necessary header files.
- Create a socket using socket function with family AF_INET, type as SOCK_DGRAM.
- Initialize server address to O using the bzero function.
- Assign the sin family to AF_INET.
- Get the server IP address and the Port number from the console.
- Using gethostbyname() function assign it to a hostent structure, and assign it to sin_addr of the server address structure.
- Obtain the command to be executed in the server from the user.
- Send the command to the server.
- Receive the output from the server and print it on the console.

Codes:

Server:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <netinet/in.h>
#include <string.h>
#include <sys/stat.h>
#include <arpa/inet.h>
#include <unistd.h>
#define MAX 1000
int main()
{
        int serverDescriptor = socket(AF_INET, SOCK_DGRAM, 0);
         int size;
        char buffer[MAX], message[] = "Command Successfully executed !";
        struct sockaddr_in clientAddress, serverAddress;
        socklen_t clientLength = sizeof(clientAddress);
        bzero(&serverAddress, sizeof(serverAddress));
        serverAddress.sin_family = AF_INET;
        serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
        serverAddress.sin_port = htons(8079);
        bind(serverDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
while (1)
{
        bzero(buffer, sizeof(buffer));
        recvfrom(serverDescriptor, buffer, sizeof(buffer), 0, (struct sockaddr *)&clientAddress,
        &clientLength);
        system(buffer);
        printf("Command Executed ... %s ", buffer);
        sendto(serverDescriptor, message, sizeof(message), 0, (struct sockaddr *)&clientAddress,
        clientLength);
}
        close(serverDescriptor);
        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>
#define MAX 1000
int main()
{
        int serverDescriptor = socket(AF_INET, SOCK_DGRAM, 0);
        char buffer[MAX], message[MAX];
        struct sockaddr_in cliaddr, serverAddress;
        socklen_t serverLength = sizeof(serverAddress);
        bzero(&serverAddress, sizeof(serverAddress));
        serverAddress.sin_family = AF_INET;
        serverAddress.sin_addr.s_addr = inet_addr("127.0.0.1");
        serverAddress.sin_port = htons(8079);
        bind(serverDescriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));
while (1)
{
        printf("\nCOMMAND FOR EXECUTION ... ");
        fgets(buffer, sizeof(buffer), stdin);
        sendto(serverDescriptor, buffer, sizeof(buffer), 0, (struct sockaddr *)&serverAddress,
        serverLength);
        printf("\nData Sent !");
        recvfrom(serverDescriptor, message, sizeof(message), 0, (struct sockaddr
        *)&serverAddress, &serverLength);
        printf("UDP SERVER : %s", message);
}
return 0;
}
```

Output:

Server:

```
RA1911026010114:~/environment/RA1911026010114/CN LAB 9/SERVER (master) $ cc server.c
RA1911026010114:~/environment/RA1911026010114/CN LAB 9/SERVER (master) $ ./a.out
Command Executed ... vi text.txt
a.out  server.c  text.txt
 Command Executed ... ls
Hi my name is Amulya
Nice to meet you.
How are you?
Have a great day.
Bye bye

 Command Executed ... cat text.txt
```

Client:

```
RA1911026010114:~/environment/RA1911026010114/CN LAB 9/CLIENT (master) $ cc client.c
RA1911026010114:~/environment/RA1911026010114/CN LAB 9/CLIENT (master) $ ./a.out

COMMAND FOR EXECUTION ... vi text.txt

Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ... ls

Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ... cat text.txt

Data Sent !UDP SERVER : Command Successfully executed !
COMMAND FOR EXECUTION ...
```

Result :

Thus the Remote Command Execution between the client and server is implemented.