COMPUTER NETWORKS AND SECURITY LABORATORY

Group B

Assignment No. 10

NAME :- OJUS P. JAISWAL

ROLL NO. :- TACO19108

YEAR AND DIV :- TE A

Ques :- Write a program using UDP Sockets to enable file transfer (Text) between two machines.

Solution :-

Program :

a) Say Hello to Each other =>

1) Server

```
// Hello UDP Server

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

#define buffsize  150

int main(void)
{
  struct sockaddr_in servaddr,clientaddr;
  char buff[buffsize],crcbuff[buffsize];
  int listenfd,connfd;
  int sin_size;

  if((listenfd=socket(AF_INET,SOCK_DGRAM,0))==-1)
          perror("Socket Creation Error.\n");
  else
          printf("Socket Created Successfully\n");
  bzero((char *) &servaddr, sizeof(servaddr));
  servaddr.sin_family=AF_INET;
```

```
 servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
 servaddr.sin_port=htons(5000);
 if(bind(listenfd, (struct sockaddr *)&servaddr,sizeof(servaddr)) == -1)
          perror("Bind Error\n");
 listen(listenfd,4);
 sin_size = sizeof(struct sockaddr_in);
 for(;;)
 {
    if(recvfrom(listenfd,buff,buffsize,0,
            (struct sockaddr *) &clientaddr, &sin_size) != buffsize)
           perror("recvfrom error\n");
 else
         printf("%s\n",buff);

}
close(connfd);
}
```

2) Client

```
//Hello UDP Client

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

#define buffsize  150

int main(void)
{
 struct sockaddr_in clientaddr;
 char buff[buffsize];
 int sockfd,n;
 if((sockfd=socket(AF_INET,SOCK_DGRAM,0))<0)
          perror("Socket Creation Error.\n");
 else
          printf("Socket Created Successfully\n");
```

```
bzero((char *) &clientaddr, sizeof(clientaddr));
clientaddr.sin_family=AF_INET;
clientaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
clientaddr.sin_port=htons(5000);
if(connect(sockfd,(struct sockaddr *) &clientaddr, sizeof(clientaddr)) < 0 )
          perror("Connect error\n");
else
        printf("Connected successfully\n");
strcpy(buff,"Hello Server\n");

if(sendto(sockfd,buff,buffsize,0,
           (struct sockaddr *) &clientaddr, sizeof(clientaddr)) != buffsize)
          perror("Send error\n");
else
        {
          printf("Buffer has sent successfully\n");
      printf("%s\n",buff);
    }

 close(sockfd);

}
```

b) File transfer =>

1) Server

```
import java.net.*;
import java.io.*;
public class UdpFSer
{
    public static void main(String args[]) throws IOException {
    byte b[] = new byte[1024];
    DatagramSocket dsoc = new DatagramSocket(9000);
    FileWriter fileWriter = new FileWriter("/home/dypiemr-/Desktop/temp.txt");
    BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
    while (true) {
       DatagramPacket dp = new DatagramPacket(b, b.length);
       dsoc.receive(dp);
       String str = new String(new String(dp.getData(), 0, dp.getLength()));
       if (str.trim().equals("END"))
          break;
       bufferedWriter.write(str);
       bufferedWriter.newLine();
    }
    bufferedWriter.close();
 System.out.println("File Send...");
 }
}
```

2) Client

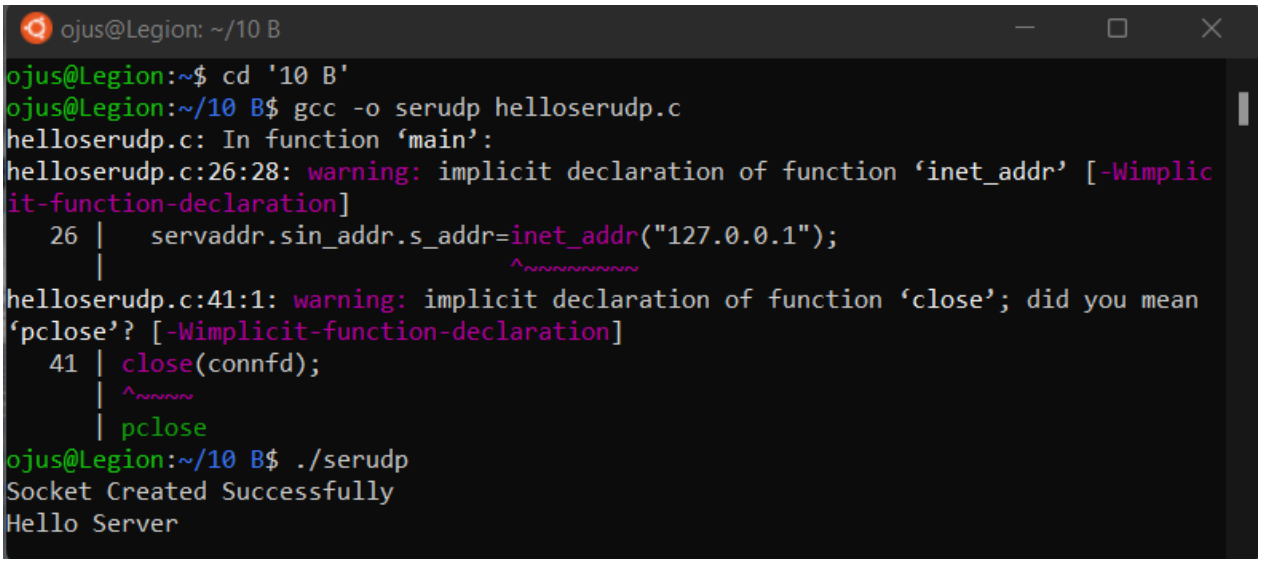
```
import java.net.*;
import java.io.*;
public class UdpFCli
{
      public static void main(String args[]) throws Exception {
```

```
      byte sdata[] = new byte[1024];
      DatagramSocket dsoc = new DatagramSocket();
      InetAddress ip = InetAddress.getByName("localhost");
      FileInputStream inputStream = new FileInputStream("/home/dypiemr-
/Desktop/temp.txt");
      int nRead = 0;
      while ((nRead = inputStream.read(sdata)) != -1) {
         dsoc.send(new DatagramPacket(sdata, sdata.length, ip, 9001));
         if (nRead == 0) {
            sdata = "END".getBytes();
            dsoc.send(new DatagramPacket(sdata, sdata.length, ip, 9001));
         }
      }
            System.out.println("File Received...");

   }

}
```

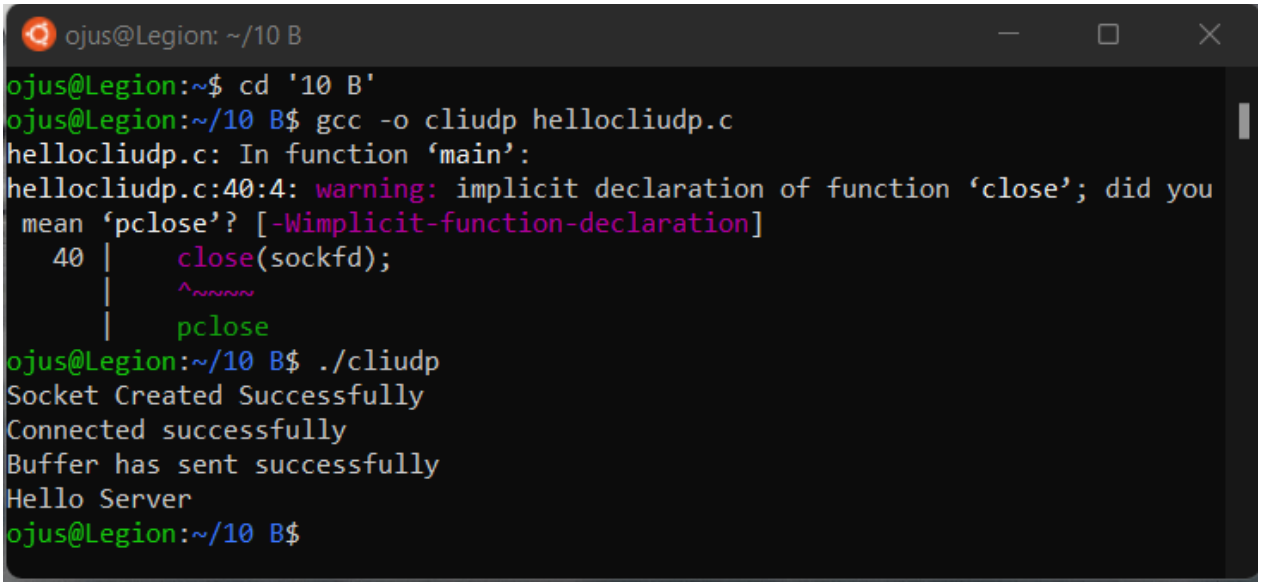
Output :

a) Say Hello to Each other =>

1) Server

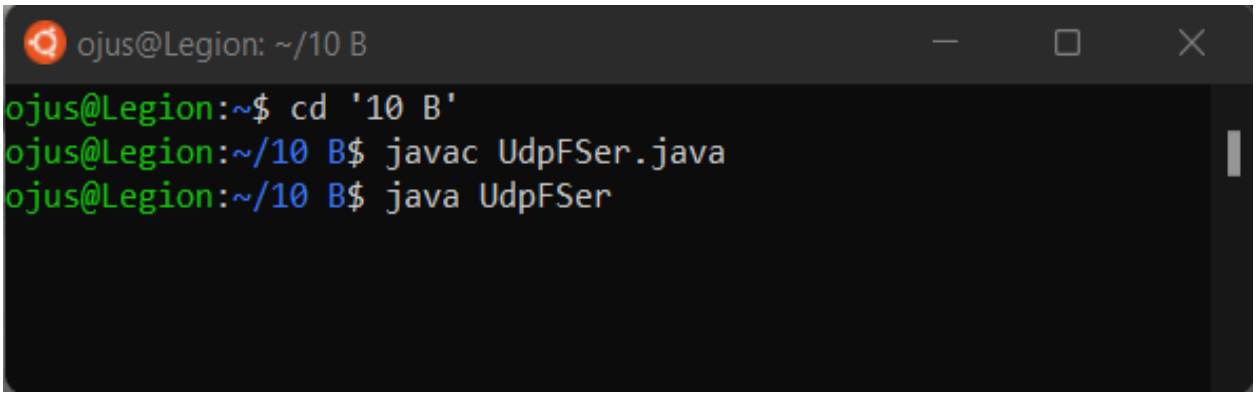


2) Client



b) File transfer =>

1) Server



2) Client