Implementation of Connection oriented concurrent service (TCP) – Reverse String

-----------Server--------

#include<sys/types.h>

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

#include<netinet/in.h>

//#define printf pf

int main(int argc,char \* argv[])

{

       int sd,bi,li,ac;

       char buffer[1024]={0};

       struct sockaddr\_in sa;

       sd=socket(AF\_INET,SOCK\_STREAM,0);

       sa.sin\_family=AF\_INET;

       sa.sin\_port=htons(atoi(argv[1]));

       sa.sin\_addr.s\_addr=inet\_addr("172.20.0.7");

               printf("socket created \n");

               printf("socked descriptor is %d \n",sd);

               bi=bind(sd,(struct sockaddr\*)&sa,sizeof(sa));

               if(bi==0)

                       {

                               printf("address binded \n");

                               li=listen(sd,5);

                               if(li==0)

                                 {

                                   printf("listen is succesful \n");

                                       int t=sizeof (sa);

                                   ac= accept(sd,(struct sockaddr \*) &sa,(socklen\_t \* )&t);

                                    if(ac>=0)

                                      {

                                        printf("accept accepted \n");

                                        printf("new file descriptor is %d \n",ac);

                                        int f=fork();

                                        if(f>0)

                                          {

                                            printf("Parent process \n");

                                          }

                                        else if(f==0)

                                          {

                                           printf("child process \n");

                                           ssize\_t rc;

                                           rc= recv(ac,buffer,1024,0);

                                           // printf("error place\n");

                                             if(rc==-1)

                                               {

                                                 printf("Receive Unsuccesful \n");

                                               }

                                             else

                                                {

                                                 printf("Receive succesful \n");

                                                 int m=strlen(buffer);

                                                 int i;

                                                 for(i=m-1;i>=0;i--)

                                                   {

                                                   printf("%c",buffer[i]);

                                                   }

                                                   printf("%n");

                                                 //printf("Buffer is %d \n",strlen(buffer));

                                                }

                                                 ssize\_t se;

                                                // char msg="Hello";

                                                 //int g=sizeof(sa);

                                                 se=send(ac,buffer,1024,0);

                                                 if(se!=-1)

                                                   {

                                                     printf("send succesful \n");

                                                   }

                                                 else

                                                    {

                                                      printf("send unsuccesful \n");

                                                    }

                                          }

                                        else

                                          {

                                           printf("child not created \n");

                                          }

                                      }

                                    else

                                      {

                                       printf("accept rejected \n");

                                      }

                                 }

                               else

                                 {

                                   printf("listen unsuccesful \n");

                                 }

                       }

}

---------------Client------------------------------

#include<sys/types.h>

#include<sys/socket.h>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

#include<netinet/in.h>

//#define printf pf

int main(int argc,char \* argv[])

{

       int sd,bi,li,ac;

       char buffer[1024]={0};

       struct sockaddr\_in sa;

       sd=socket(AF\_INET,SOCK\_STREAM,0);

       sa.sin\_family=AF\_INET;

       sa.sin\_port=htons(atoi(argv[1]));

       sa.sin\_addr.s\_addr=inet\_addr("172.20.0.7");

       printf("socket created \n");

       printf("socked descriptor is %d \n",sd);

       int conn;

       conn=connect(sd,(struct sockaddr \*)&sa,sizeof(sa));

       if(conn==0)

       {

               printf("connection is established \n");

       }

       else

       {

               printf("connection error");

               return 0;

       }

               ssize\_t se;

               char msg[5]="Hello";

               //scanf("%s",&msg);

               se=send(sd,msg,5,0);

               if(se>=0)

               {

                       printf("send succesful \n");

               }

               else

               {

                       printf("send unsuccesful \n");

 }

                       ssize\_t rc;

                       rc=recv(sd,buffer,1024,0);

                       if(rc>=0)

                        {

                          printf("Receive Succesful \n");

                          printf("Message is %s \n",buffer);

                        }

                       else

                        {

                          printf("Receive unsuccesful \n");

                        }

       }

Implementation of Connection oriented concurrent service (TCP).

**server.c**

**#include<stdio.h>**

**#include<netinet/in.h>**

**#include<unistd.h>**

**#include<sys/types.h>**

**#include<sys/socket.h>**

**#include<string.h>**

**int main()**

**{**

**int sfd,afd;**

**struct sockaddr\_in local,addr;**

**char \*buf;**

**sfd=socket(AF\_INET,SOCK\_STREAM,0);**

**if(sfd==-1)**

**{**

**printf("socket error\n");**

**}**

**local.sin\_family = AF\_INET;**

**local.sin\_addr.s\_addr = inet\_addr("172.20.0.7");**

**local.sin\_port = htons(1078);**

**if(bind(sfd,(struct sockaddr\*)&local,sizeof(local))<0)**

**{**

**printf("bind error\n");**

**}**

**if(listen(sfd,5)<0)**

**{**

**printf("listen error!\n");**

**}**

**int l=sizeof(addr);**

**afd=accept(sfd,(struct sockaddr\*)&addr,&l);**

**if(afd==-1)**

**{**

**printf("error!\n");**

**}**

**else**

**{**

**close(sfd);**

**printf("Connection established\n");**

**int pd=fork();**

**if(pd==0)**

**{**

**read(afd,buf,100);**

**printf("Recieved message:%s\n",buf);**

**}**

**}**

**close(afd);**

**return 0;**

**}**

**client.c**

**#include<stdio.h>**

**#include<netinet/in.h>**

**#include<unistd.h>**

**#include<sys/types.h>**

**#include<sys/socket.h>**

**#include<string.h>**

**int main()**

**{**

**int sfd,con;**

**struct sockaddr\_in local;**

**sfd=socket(AF\_INET,SOCK\_STREAM,0);**

**local.sin\_family = AF\_INET;**

**local.sin\_addr.s\_addr = inet\_addr("172.20.0.7");**

**local.sin\_port = htons(1078);**

**con=connect(sfd,(struct sockaddr\*)&local,(socklen\_t)sizeof(local));**

**char str[]="HI";**

**write(sfd,str,sizeof(str));**

**close(sfd);**

**return 0;**

**}**

Implementation of Connectionless Iterative time service (UDP).

**time\_server.c**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<time.h>**

**#include<sys/socket.h>**

**#include<sys/types.h>**

**#include<string.h>**

**#include<errno.h>**

**#include<netinet/in.h>**

**#define PORT 6008**

**int main(int argc, char\*argv[])**

**{**

**int sfd;**

**time\_t ct;**

**sfd=socket(AF\_INET,SOCK\_DGRAM,0);**

**struct sockaddr\_in local,address;**

**//SOCKET**

**if(sfd==0)**

**{**

**printf("Socket Creation Unsuccessful\n");**

**}**

**else**

**{**

**printf("Socket Created Successfully.\nSFD : %d\n",sfd);**

**}**

**local.sin\_family = AF\_INET;**

**local.sin\_addr.s\_addr = INADDR\_ANY;**

**local.sin\_port = htons(PORT);**

**//BIND**

**int b=bind(sfd,(struct sockaddr \*)&local,sizeof(local));**

**if(b==-1)**

**{**

**printf("Bind Unsuccessful\n");**

**}**

**else**

**{**

**printf("Bind Successful\n");**

**}**

**char buffer[1024]={0};**

**int t=sizeof(address);**

**//RECVFROM**

**int a1 =recvfrom(sfd,buffer,1024,0,(struct sockaddr \*)&address,(socklen\_t \*)&t);**

**//ct=time(NULL);**

**if(a1==-1)**

**{**

**printf("Receive Unsuccessful\n");**

**}**

**else**

**{**

**printf("Receive Successful\n");**

**printf("Message : %s\n",buffer);**

**ct = time(NULL);**

**//char s=ctime(&ct);**

**if((strcmp("TIME",buffer))==0)**

**{**

**//printf("Current Time : %s\n",ctime(&ct));**

**sprintf(buffer,"%s",ctime(&ct));**

**//SENDTO**

**int b1=sendto(sfd,buffer,sizeof(buffer),0,(struct sockaddr \*)&address,sizeof(address));**

**if(b1==-1)**

**{**

**printf("Send Unsuccessful\n");**

**}**

**else**

**{**

**printf("Send Successful\n");**

**}**

**}**

**else**

**{**

**printf("Strings not equal\n");**

**}**

**}**

**//CLOSE**

**close(sfd);**

**}**

**time\_client.c**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<time.h>**

**#include<sys/socket.h>**

**#include<sys/types.h>**

**#include<string.h>**

**#include<errno.h>**

**#include<netinet/in.h>**

**#define PORT 6008**

**int main(int argc, char\*argv[])**

**{**

**int sfd;**

**time\_t ct;**

**//SOCKET CREATION**

**sfd=socket(AF\_INET,SOCK\_DGRAM,0);**

**struct sockaddr\_in local,remote;**

**if(sfd==0)**

**{**

**printf("Socket Creation Unsuccessful\n");**

**}**

**else**

**{**

**printf("Socket Created Successfully.\nSFD : %d\n",sfd);**

**}**

**local.sin\_family = AF\_INET;**

**local.sin\_addr.s\_addr = INADDR\_ANY;**

**local.sin\_port =atoi(argv[1]);**

**remote.sin\_family = AF\_INET;**

**remote.sin\_addr.s\_addr = INADDR\_ANY;**

**remote.sin\_port = htons(PORT);**

**//BIND**

**int b=bind(sfd,(struct sockaddr \*)&local,sizeof(local));**

**if(b==-1)**

**{**

**printf("Bind Unsuccessful\n");**

**}**

**else**

**{**

**printf("Bind Successful\n");**

**}**

**char buffer[1024]={0};**

**char s[20],buf[20];**

**strcpy(s,"TIME");**

**int r=sizeof(remote);**

**//SENDTO**

**int b1=sendto(sfd,s,sizeof(s),0,(struct sockaddr\*)&remote,r);**

**if(b1==-1)**

**{**

**printf("Send Unsuccessful\n");**

**}**

**else**

**{**

**printf("Send Successful\n");**

**}**

**int l=sizeof(local);**

**//RECVFROM**

**int a1=recvfrom(sfd,buffer,sizeof(buffer),0,(struct sockaddr\*)&local,(socklen\_t \*)&l);**

**if(a1==-1)**

**{**

**printf("Receive Unsuccessful\n");**

**}**

**else**

**{**

**printf("Receive Successful\n");**

**printf("%s\n",buffer);**

**}**

**//CLOSE**

**close(sfd);**

**}**

Implementation of Select system call

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<errno.h>**

**#include<sys/types.h>**

**#include<sys/time.h>**

**int main(int argc, char \*argv[])**

**{**

**static struct timeval timeout;**

**if(argc!=3)**

**{**

**printf("\n Usage : %s <seconds> <microseconds>",argv[0]);**

**exit(0);**

**}**

**timeout.tv\_sec=atol(argv[1]);**

**timeout.tv\_usec=atol(argv[2]);**

**if(select(0,(fd\_set \*) 0,(fd\_set \*) 0,(fd\_set \*) 0,&timeout)<0)**

**{ perror("select"); exit(0); }**

**return 0;**

**}**

Implementation of getsockopt(), setsockopt() system calls.

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<errno.h>**

**#include<string.h>**

**#include<sys/types.h>**

**#include<sys/socket.h>**

**#include<netinet/in.h>**

**#include<netinet/tcp.h>**

**int main()**

**{**

**int sockfd,maxseg,sendbuff,optlen;**

**sockfd = socket(AF\_INET,SOCK\_STREAM,0);**

**if( sockfd < 0 )**

**{ perror("socket"); exit(0); }**

**optlen=sizeof(maxseg);**

**if( getsockopt(sockfd,IPPROTO\_TCP,TCP\_MAXSEG,(char \*)&maxseg,&optlen) < 0 )**

**{ perror("getsockopt1"); exit(0); }**

**printf("\n TCP maxseg=%d",maxseg);**

**sendbuff=12324;**

**if(setsockopt(sockfd,SOL\_SOCKET,SO\_SNDBUF,(char \*)&sendbuff,sizeof(sendbuff))<0)**

**{ perror("setsockopt"); exit(0); }**

**optlen=sizeof(sendbuff);**

**if(getsockopt(sockfd,SOL\_SOCKET,SO\_SNDBUF,(char \*)&sendbuff,&optlen)<0)**

**{ perror("getsockopt2"); exit(0); }**

**printf("\n Send Buffer size=%d",sendbuff);**

**return 0;**

**}**

Implementation of getpeername() system call.

**Getpeername\_server.c**

**#include<stdio.h>**

**#include<unistd.h>**

**#include<string.h>**

**#include<stdlib.h>**

**#include<sys/types.h>**

**#include<sys/socket.h>**

**#include<netinet/in.h>**

**#include<netinet/tcp.h>**

**#include<errno.h>**

**int main( )**

**{**

**int s, s2, t, len;**

**struct sockaddr\_in local,rem;**

**char str[100];**

**s = socket(AF\_INET,SOCK\_STREAM,0);**

**if(s==-1)**

**{**

**perror("socket");**

**exit(1);**

**}**

**bzero((char\*)&local,sizeof(local));**

**local.sin\_family=AF\_INET;**

**local.sin\_port=htons(9247);**

**local.sin\_addr.s\_addr=inet\_addr("172.20.0.7");**

**if(bind(s,(struct sockaddr\*)&local,sizeof(local))==-1)**

**{ perror("bind");**

**exit(1);**

**}**

**if(listen(s,5)==-1)**

**{**

**perror("listen");**

**exit(1);**

**}**

**int done,n;**

**//printf("waiting for a connection....\n");**

**t=sizeof(rem);**

**s2=accept(s,(struct sockaddr\*)&rem,&t);**

**if(s2==-1)**

**{**

**perror("accept");**

**exit(1);**

**}**

**close(s2);**

**return 0;**

**}**

**Getpeername\_client.c**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<errno.h>**

**#include<string.h>**

**#include<sys/types.h>**

**#include<sys/socket.h>**

**#include<netinet/in.h>**

**int main()**

**{**

**int s;**

**struct sockaddr\_in server,addr;**

**socklen\_t len;**

**int port;**

**char ipstr[INET\_ADDRSTRLEN];**

**s=socket(AF\_INET,SOCK\_STREAM,0);**

**if(s==-1)**

**{**

**perror("socket");**

**exit(1);**

**}**

**server.sin\_family=AF\_INET;**

**inet\_aton("172.20.0.7",&server.sin\_addr);**

**server.sin\_port=htons(9247);**

**if(connect(s,(struct sockaddr\*)&server,sizeof(server))<0)**

**{**

**perror("connect");**

**exit(0);**

**}**

**len=sizeof(addr);**

**getpeername(s,(struct sockaddr \*)&addr,&len);**

**inet\_ntop(AF\_INET,&addr.sin\_addr,ipstr,sizeof(ipstr));**

**printf("peer IP address: %s\n", ipstr);**

**printf("peer port: %d\n",ntohs(addr.sin\_port));**

**return 0;**

**}**

Implementation of remote command execution using socket system calls.

**Remote\_server.c**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<sys/types.h>**

**#include<sys/socket.h>**

**#include<string.h>**

**#include<netinet/in.h>**

**#define MAX 1024**

**int main(int argc, char \*argv[])**

**{**

**int sockfd,addrlen,new,n;**

**char buff1[MAX];**

**struct sockaddr\_in serv\_addr,peer;**

**sockfd=socket(AF\_INET,SOCK\_STREAM,0);**

**if( sockfd == -1)**

**{ perror("Error in Server");**

**exit(1);**

**}**

**serv\_addr.sin\_family=AF\_INET;**

**serv\_addr.sin\_port=htons(atoi(argv[1]));**

**serv\_addr.sin\_addr.s\_addr=inet\_addr("172.20.0.7");**

**if( bind(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr)) < 0 )**

**{ perror("Error in Server ");**

**exit(1);**

**}**

**if( listen(sockfd,5) < 0 )**

**{ perror("Error in Server ");**

**exit(1);**

**}**

**for( ; ; )**

**{**

**if( (new=accept(sockfd,(struct sockaddr \*)&peer,&addrlen)) < 0 )**

**{ perror("Error in Server");**

**exit(1);**

**}**

**if( (fork()) == 0 )**

**{**

**for( ; ; )**

**{**

**if( (n=read(new,buff1,MAX)) < 0 )**

**{ perror("Error in Server ");**

**exit(1); }**

**if( n == 0 )**

**break;**

**buff1[n]='\0'; // fflush(stdout);**

**if( !strcmp("exit",buff1) )**

**{ printf("\a\a server is exiting");**

**exit(1);**

**}**

**close(1);**

**if(dup(new)<0)**

**printf("\nserver: dup system call failure");**

**fflush(stdout);**

**system(buff1);**

**if((write(new,"\n",1))<0)**

**perror("write error");**

**}**

**}**

**close(new);**

**}**

**}**

**Remote\_client.c**

**#include <stdio.h>**

**#include <sys/types.h>**

**#include <sys/socket.h>**

**#include <netinet/in.h>**

**#include<stdlib.h>**

**#define MAX 1024**

**int main(int argc,char \*argv[])**

**{**

**int sockfd,addr\_len,n;**

**char buff1[MAX],buff2[MAX];**

**struct sockaddr\_in serv\_addr;**

**sockfd=socket(AF\_INET,SOCK\_STREAM,0);**

**if(sockfd==-1)**

**{ perror("Error in Client:");**

**exit(1);**

**}**

**serv\_addr.sin\_family=AF\_INET;**

**serv\_addr.sin\_port=htons(atoi(argv[1]));**

**serv\_addr.sin\_addr.s\_addr=inet\_addr("172.20.0.7");**

**if((connect(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr)))<0)**

**{ perror("Error in Client: ");**

**exit(1);**

**}**

**fflush(stdout);**

**for( ; ; )**

**{**

**printf("\nEnter the command string: ");**

**fflush(stdout);**

**if( (n=read(0,buff1,MAX+1)) < 0 )**

**{ perror("Error in Client: ");**

**exit(1);**

**}**

**buff1[n]='\0';**

**if( (n=write(sockfd,buff1,n+1)) < 0 )**

**perror("Error in Client: ");**

**if( (bcmp("exit",buff1,4)) == 0 )**

**{ printf("\a\a Client is exiting...\n");**

**exit(1);**

**}**

**sleep(1);**

**if((n=read(sockfd,buff2,MAX))<0)**

**{ perror("Error in Client: ");**

**exit(1);**

**}**

**buff2[n]='\0';**

**if(write(1,buff2,n+1)<0)**

**perror("Client:Write by client ot the screen");**

**}**

**return 0;**

**}**

**Output:**

**(Server side)**

[it16-107@localhost np]$ vi systemserver.c

[it16-107@localhost np]$ gcc systemserver.c -o ser

(Compile client now)

(After compiling there run this)

[it16-107@localhost np]$ ./ser 6666

write error: Connection reset by peer

**output:**

**(Client side)**

login as: it16-107

it16-107@172.20.0.7's password:

Last login: Mon Aug 4 10:33:59 2008 from 172.20.7.181

[it16-107@localhost ~]$ cd np

[it16-107@localhost np]$ clear

[it16-107@localhost np]$ vi systemclient.c

[it16-107@localhost np]$ gcc systemclient.c -o cli

(Run this after running client code)

[it16-107@localhost np]$ ./cli 6666

Enter the command string: ls

a

a.out

c

cli

echoclient.c

echoserver.c

getpeernameclient.c

getpeernameserver.c

getsetsockopt.c

rc

remoteclient.c

remoteserver.c

rs

RSA.class

rsa.java

RSA.java

s

select.c

ser

sl

systemclient.c

systemserver.c

tc

tcpclient.c

tcpconcurrentclient.c

tcpconcurrentserver.c

tcpserver.c

ts

udoclient.c

udpserver.c

Enter the command string: pwd

/home/it16/it16-107/np

Enter the command string: pstree

init-+-NetworkManager

|-abrt-dump-oops

|-abrtd

|-acpid

|-atd

|-automount---4\*[{automount}]

|-avahi-daemon---avahi-daemon

|-bonobo-activati---{bonobo-activat}

|-certmonger

|-console-kit-dae---63\*[{console-kit-da}]

|-crond

|-cupsd

|-2\*[dbus-daemon---{dbus-daemon}]

|-dbus-launch

|-devkit-power-da

|-dnsmasq

|-gconfd-2

|-gdm-binary---gdm-simple-slav-+-Xorg

| |-gdm-session-wor

| `-gnome-session-+-at-s pi-registry

| |-gdm- simple-gree

| |-gnom e-power-man

| |-meta city

| |-polk it-gnome-au

| `-{gno me-session}

|-gnome-settings----{gnome-settings}

|-gvfsd

|-hald---hald-runner-+-hald-addon-acpi

|

Enter the command string: exit

Client is exiting...

**SIMPLE HTTP SERVER PROGRAM**

**import** java.io.IOException;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**import** java.util.Date;

**public** **class** SimpleHTTPServer

{

**public** **static** **void** main(String args[]) **throws** IOException

{

ServerSocket server = **new** ServerSocket(8080);

System.***out***.println("Listening for connection on port 8080 ....");

**while** (**true**)

{

**try** (Socket socket = server.accept())

{

Date today = **new** Date();

String httpResponse = "HTTP/1.1 200 OK\r\n\r\n" + today;

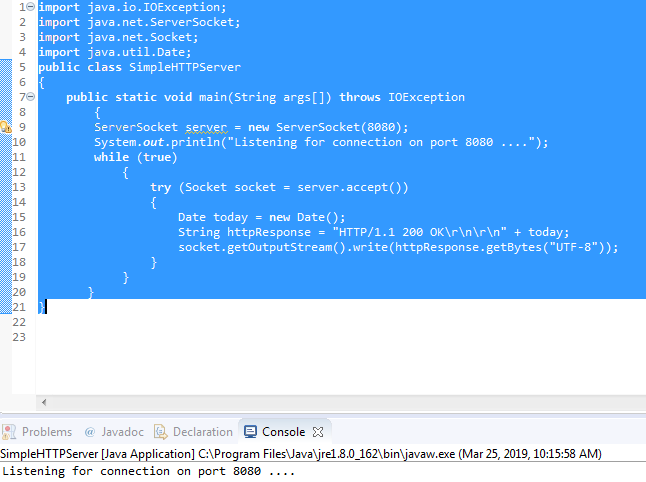
socket.getOutputStream().write(httpResponse.getBytes("UTF-8"));

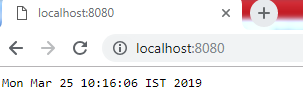
}

}

}

}





**FTP Server**

**package** ftpserver;

**import** java.io.DataInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**public** **class** ftpserver **extends** Thread {

**private** ServerSocket ss;

/\* creating socket for communication\*/

**public** ftpserver(**int** port) {

**try** {

ss = **new** ServerSocket(port);

} **catch** (IOException e) {

e.printStackTrace();

}

}

/\* thread.run() method basically used to run thread class\*/

**public** **void** run() {

**while** (**true**) {

**try** {

Socket clientSock = ss.accept();// to bind the ip adress and port no

saveFile(clientSock);

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

/\* file handling or input calling methods we r using\*/

**private** **void** saveFile(Socket clientSock) **throws** IOException {

DataInputStream dis = **new** DataInputStream(clientSock.getInputStream());

FileOutputStream fos = **new** FileOutputStream("sample.txt");

**byte**[] buffer = **new** **byte**[4096];

**int** filesize = 777777; // insert the file size depend upon the file

**int** read = 0;

**int** totalRead = 0;

**int** remaining = filesize;

**while**((read = dis.read(buffer, 0, Math.*min*(buffer.length, remaining))) > 0) {

totalRead += read;

remaining -= read;

System.***out***.println("read " + totalRead + " bytes.");

fos.write(buffer, 0, read);

}

fos.close();

dis.close();

}

**public** **static** **void** main(String[] args) {

ftpserver fs = **new** ftpserver(4000);

fs.start();

}

}

**FTP CLIENT**

**package** ftpclient;

/\* ftp client program \*/

**import** java.io.DataOutputStream;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.net.Socket;

**public** **class** ftpclient {

**private** Socket s;

/\*accepting ip add $ port no $ file name\*/

**public** ftpclient(String host, **int** port, String file) {

**try** {

s = **new** Socket(host, port);

sendFile(file);

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** sendFile(String file) **throws** IOException {

DataOutputStream dos = **new** DataOutputStream(s.getOutputStream());

FileInputStream fis = **new** FileInputStream(file);

**byte**[] buffer = **new** **byte**[4096];

**while** (fis.read(buffer) > 0) {

dos.write(buffer);

}

fis.close();

dos.close();

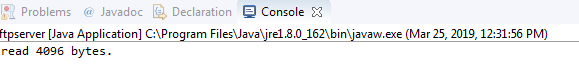
}

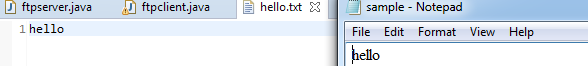
**public** **static** **void** main(String[] args) {

ftpclient fc = **new** ftpclient("localhost", 4000, "hello.txt");//arguments to pass about file

}

}





DVR :

import java.io.\*;  
public class DVR   
{  
 static int graph[][];  
 static int via[][];  
 static int rt[][];  
 static int v;  
 static int e;  
  
 public static void main(String args[]) throws IOException  
 {  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
   
 System.out.println("Please enter the number of Vertices: ");  
 v = Integer.parseInt(br.readLine());  
   
 System.out.println("Please enter the number of Edges: ");  
 e = Integer.parseInt(br.readLine());  
   
 graph = new int[v][v];  
 via = new int[v][v];  
 rt = new int[v][v];  
 for(int i = 0; i < v; i++)  
 for(int j = 0; j < v; j++)  
 {  
 if(i == j)  
 graph[i][j] = 0;  
 else  
 graph[i][j] = 9999;  
 }  
   
 for(int i = 0; i < e; i++)  
 {  
 System.out.println("Please enter data for Edge " + (i + 1) + ":");  
 System.out.print("Source: ");  
 int s = Integer.parseInt(br.readLine());  
 s--;  
 System.out.print("Destination: ");  
 int d = Integer.parseInt(br.readLine());  
 d--;  
 System.out.print("Cost: ");  
 int c = Integer.parseInt(br.readLine());  
 graph[s][d] = c;  
 graph[d][s] = c;  
 }  
   
 dvr\_calc\_disp("The initial Routing Tables are: ");  
   
 System.out.print("Please enter the Source Node for the edge whose cost has changed: ");  
 int s = Integer.parseInt(br.readLine());  
 s--;  
 System.out.print("Please enter the Destination Node for the edge whose cost has changed: ");  
 int d = Integer.parseInt(br.readLine());  
 d--;  
 System.out.print("Please enter the new cost: ");  
 int c = Integer.parseInt(br.readLine());  
 graph[s][d] = c;  
 graph[d][s] = c;  
   
 dvr\_calc\_disp("The new Routing Tables are: ");  
 }  
   
 static void dvr\_calc\_disp(String message)  
 {  
 System.out.println();  
 init\_tables();  
 update\_tables();  
 System.out.println(message);  
 print\_tables();  
 System.out.println();  
 }  
   
 static void update\_table(int source)  
 {  
 for(int i = 0; i < v; i++)  
 {  
 if(graph[source][i] != 9999)  
 {  
 int dist = graph[source][i];  
 for(int j = 0; j < v; j++)  
 {  
 int inter\_dist = rt[i][j];  
 if(via[i][j] == source)  
 inter\_dist = 9999;  
 if(dist + inter\_dist < rt[source][j])  
 {  
 rt[source][j] = dist + inter\_dist;  
 via[source][j] = i;  
 }  
 }  
 }  
 }  
 }  
   
 static void update\_tables()  
 {  
 int k = 0;  
 for(int i = 0; i < 4\*v; i++)  
 {  
 update\_table(k);  
 k++;  
 if(k == v)  
 k = 0;  
 }  
 }  
   
 static void init\_tables()  
 {  
 for(int i = 0; i < v; i++)  
 {  
 for(int j = 0; j < v; j++)  
 {  
 if(i == j)  
 {  
 rt[i][j] = 0;  
 via[i][j] = i;  
 }  
 else  
 {  
 rt[i][j] = 9999;  
 via[i][j] = 100;  
 }  
 }  
 }  
 }  
   
 static void print\_tables()  
 {  
 for(int i = 0; i < v; i++)  
 {  
 for(int j = 0; j < v; j++)  
 {  
 System.out.print("Dist: " + rt[i][j] + " ");  
 }  
 System.out.println();  
 }  
 }  
   
}  
output:-  
Please enter the number of Vertices:   
4  
Please enter the number of Edges:   
5  
Please enter data for Edge 1:  
Source: 1  
Destination: 2  
Cost: 1  
Please enter data for Edge 2:  
Source: 1  
Destination: 3  
Cost: 3  
Please enter data for Edge 3:  
Source: 2  
Destination: 3  
Cost: 1  
Please enter data for Edge 4:  
Source: 2  
Destination: 4  
Cost: 1  
Please enter data for Edge 5:  
Source: 3  
Destination: 4  
Cost: 4  
  
The initial Routing Tables are:   
Dist: 0 Dist: 1 Dist: 2 Dist: 2   
Dist: 1 Dist: 0 Dist: 1 Dist: 1   
Dist: 2 Dist: 1 Dist: 0 Dist: 2   
Dist: 2 Dist: 1 Dist: 2 Dist: 0   
  
Please enter the Source Node for the edge whose cost has changed: 2  
Please enter the Destination Node for the edge whose cost has changed: 4  
Please enter the new cost: 10  
  
The new Routing Tables are:   
Dist: 0 Dist: 1 Dist: 2 Dist: 6   
Dist: 1 Dist: 0 Dist: 1 Dist: 5   
Dist: 2 Dist: 1 Dist: 0 Dist: 4   
Dist: 6 Dist: 5 Dist: 4 Dist: 0   
--------------------------------

RSA:

import java.io.DataInputStream;  
import java.io.IOException;  
import java.math.BigInteger;  
import java.util.Random;  
   
public class RSA  
{  
 private BigInteger p;  
 private BigInteger q;  
 private BigInteger N;  
 private BigInteger phi;  
 private BigInteger e;  
 private BigInteger d;  
 private int bitlength = 1024;  
 private Random r;  
   
 public RSA()  
 {  
 r = new Random();  
 p = BigInteger.probablePrime(bitlength, r);  
 q = BigInteger.probablePrime(bitlength, r);  
 N = p.multiply(q);  
 phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));  
 e = BigInteger.probablePrime(bitlength / 2, r);  
 while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)  
 {  
 e.add(BigInteger.ONE);  
 }  
 d = e.modInverse(phi);  
 }  
   
 public RSA(BigInteger e, BigInteger d, BigInteger N)  
 {  
 this.e = e;  
 this.d = d;  
 this.N = N;  
 }  
   
 @SuppressWarnings("deprecation")  
 public static void main(String[] args) throws IOException  
 {  
 RSA rsa = new RSA();  
 DataInputStream in = new DataInputStream(System.in);  
 String teststring;  
 System.out.println("Enter the plain text:");  
 teststring = in.readLine();  
 System.out.println("Encrypting String: " + teststring);  
 System.out.println("String in Bytes: "  
 + bytesToString(teststring.getBytes()));  
 // encrypt  
 byte[] encrypted = rsa.encrypt(teststring.getBytes());  
 // decrypt  
 byte[] decrypted = rsa.decrypt(encrypted);  
 System.out.println("Decrypting Bytes: " + bytesToString(decrypted));  
 System.out.println("Decrypted String: " + new String(decrypted));  
 }  
   
 private static String bytesToString(byte[] encrypted)  
 {  
 String test = "";  
 for (byte b : encrypted)  
 {  
 test += Byte.toString(b);  
 }  
 return test;  
 }  
   
 // Encrypt message  
 public byte[] encrypt(byte[] message)  
 {  
 return (new BigInteger(message)).modPow(e, N).toByteArray();  
 }  
   
 // Decrypt message  
 public byte[] decrypt(byte[] message)  
 {  
 return (new BigInteger(message)).modPow(d, N).toByteArray();  
 }  
}

**SMTP:**

import java.util.Properties;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class Smtpmail

{

public static void main(String[] args) {

// Recipient's email ID needs to be mentioned.

String to = "";//put your sending mail id

// Sender's email ID needs to be mentioned

String from = "";//change accordingly

final String username = "";//keep as it is

final String password = "";//keep as it is

// Assuming you are sending email through relay.jangosmtp.net

String host = "smtp.gmail.com";

Properties props = new Properties();

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.smtp.host", host);

props.put("mail.smtp.port", "587");

// Get the Session object.(to maintain session and to get object)

Session session = Session.getInstance(props,

new javax.mail.Authenticator() {

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(username, password);

}

});

try {

// Create a default MimeMessage object.

Message message = new MimeMessage(session);

// Set From:

message.setFrom(new InternetAddress(from));

// Set To:

message.setRecipients(Message.RecipientType.TO,

InternetAddress.parse(to));

// Set Subject

message.setSubject("Testing Subject");

// The actual message

message.setText("Hello, this is sample to check mail "

+ "from ith2 cbit 3/4 ");

// Send message

Transport.send(message);

System.out.println("Sent message successfully....");

} catch (MessagingException e) {

throw new RuntimeException(e);

}

}

}

Output:

1. Open eclipse or netbeans, if it is netbeans. Follow these steps:

1.File->New Project->(S elect category as Java and Projects as Java application)-> Give project name->Finish

2.Right click on the created project->Properties->Libraries->add jar files downloaded from the following links and the jar files are present in these folders under lib folder.

mail.jar:

http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-eeplat-419426.html#javamail-1.4.5-oth-JPR

activation.jar:

<http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-plat-419418.html#jaf-1.1.1-fcs-oth-JPR>

3.Open the link and turn on less secure apps authentication:

https://www.google.com/settings/security/lesssecureapps

.Now run the above code

(Note that the username is the from email address and obviously from address, to address and sender’s password should be given to the strings in the above program before running)