/\*--------------------------------------------------------

1. Vivek Patel / June 2, 2019:

2. Java build 1.8.0\_201

3. Precise command-line compilation examples / instructions:

> javac \*.java

4. Precise examples / instructions to run this program:

In separate shell windows:

> java AsyncJokeServer

> java AsyncJokeClient

> java AsyncJokeAdminClient

All acceptable commands are displayed on the various consoles.

This runs across machines, in which case you have to pass the IP address of

the server to the clients. For exmaple, if the server is running at

140.192.1.22 then you would type:

> java JokeClient 140.192.1.22

> java JokeClientAdmin 140.192.1.22

5. List of files needed for running the program.

a. JokeServer.java

b. JokeClient.java

c. JokeClientAdmin.java

5. Notes:

AdminLooper and implementation in main() method comes from Prof. Elliot’s code:

http://condor.depaul.edu/elliott/435/hw/programs/joke/joke-threads.html

AsyncJokeClient takes input from the user. First, it asks for the name of the user, then it asks for a simple

<enter> to receive a joke or proverb. Client will then send the request to the AsyncJokeServer. Client will listen

on a UDP server for a response. While waiting for a response, the client will as the user to input 2 numbers with a

space separating the two. Once the Client recieves a joke/proverb from the UDP server, it will store the message and

update its state. The Client will only print a message after completing a sum.

AsyncJokeServer takes input from the AsyncJokeClient and, in a seperate port, AsyncJokeAdminClient. The request from

the client is processed by checking the current output mode (jokes or proverbs) then, before packaging and sending

data to the UDP server, the thread will sleep for 40 seconds. After the 40s, the Server will pacage the data that will

be sent to the UDP server.

AsyncJokeAdminClient is the same as the previous iteration. Takes command inputs from the user and sends them to the server.

Admin can change the server's output mode or close the Server.

The state is stored on the Client using 2 separate Lists. It is very sloppy, but it worked for me.

The Client makes a list of integers, which is then copied to 2 separate lists: one for jokes and

one for proverbs. The state lists are then shuffled. We take the first of both lists and send it to

the Server. The Server takes these indexes and picks the one corresponding to the mode and sends it

to the appropriate output method. Then the joke/proverb that has the same index as the Client's is

sent along with the index and a large number back to the client. The Client then processes this by

printing out the joke/proverb, removing the index from the temp list that it came from (which is

determined by comparing the index and the large number) and then checking if the temp list is empty.

If the temp list is empty, then the appropriate cycle completion is annouced and the temp list is

repopulated and shuffled using the master list.

Help for learning UDP server basics:

https://systembash.com/a-simple-java-udp-server-and-udp-client/

Jokes/Proverbs credits:

JA: https://www.reddit.com/r/Jokes/comments/7he12e/i\_hope\_elon\_musk\_never\_gets\_involved\_in\_a\_scandal/

JB: https://www.reddit.com/r/Jokes/comments/7v0je6/my\_girlfriend\_is\_like\_the\_square\_root\_of\_100/

JC: https://www.reddit.com/r/Jokes/comments/6zukvv/i\_invited\_my\_girlfriend\_to\_go\_to\_the\_gym\_with\_me/

JD: https://www.reddit.com/r/Jokes/comments/basb7p/as\_i\_handed\_my\_dad\_his\_50th\_birthday\_card\_he/

PA: http://www.goldenproverbs.com/au\_confucius.html

PB: http://www.goldenproverbs.com/au\_plato.html

PC: http://www.goldenproverbs.com/au\_gandhi.html

PD: http://www.goldenproverbs.com/au\_verne.html

----------------------------------------------------------\*/

import java.io.\*;

import java.net.\*;

import java.util.\*;

// Communicates with the client, recieving 2 index numbers and a name,and sends either a joke or proverb,

//depending on the mode set by the Admin, joke by default. Utilizes a UDP connection to send the joke/proverb

//to the client.

class ClientWorker extends Thread {

Socket sock;

ClientWorker (Socket s) {sock = s;} // constructor for the Client Worker

public void run() { //the main code for conversation with a client.

//PrintStream out = null;

BufferedReader in = null;

int udpPort;

//initialize joke and proverb lists

ArrayList<String> jokes = new ArrayList<String>(4);

ArrayList<String> proverbs = new ArrayList<String>(4);

InetAddress udpAddress = sock.getInetAddress(); // grabs the IP Address

StringBuilder toSend = new StringBuilder(); // creates a StringBuilder object to be sent to the client

try { // tries to create an input reader and a Datagram Socket

in = new BufferedReader(new InputStreamReader (sock.getInputStream()));

//out = new PrintStream(sock.getOutputStream());

DatagramSocket dataSock = new DatagramSocket();

try { // tries to read the input from the client, process it, and send the output to the client

// Initializes variables needed by the ClientWorker to process the input and prepare for output

// indexJ and indexP refer to the saved state stored on the client.

String tempID;

String name;

String mode;

tempID = in.readLine(); // the input from the client, containing a string of a name, 2 index values, and the UDP port the Server needs to connect to

String id[] = tempID.split(", "); // splits the input into a String array so the input can be set to the appropriate variables

name = id[0]; // this is the name the user entered into the Client

//Creates temporary strings from the input that are then converted to integers to be used as index

//values to reference the joke and proverb lists

String inJTemp = id[1];

int indexJ = Integer.parseInt(inJTemp);

String inPTemp = id[2];

int indexP = Integer.parseInt(inPTemp);

String udpTemp = id[3]; // pulls the udp port from the input

udpPort = Integer.parseInt(udpTemp); // converts the udp port string to an int

// Creates the lists for the jokes and proverbs. They are in 2 separate lists.

jokeBuilder(name, jokes);

proverbBuilder(name, proverbs);

// checks the current mode and then sends the corresponding list, index number, and

//PrintStream to create and send the output joke/proverb. Prints on the server as to

//what the server is sending and to whom.

mode = AdminWorker.mode;

Thread.sleep(40000); // the thread will sleep for 40 seconds before returning to the task at hand.

// this is to give the client time to interact with the user (in this case requesting 2 numbers and summing them)

if (mode == "joke") { // if the mode is set to "joke," then send a joke

System.out.println("Sending joke to " + name); // prints to the console who will recieve the joke

sendJoke(jokes, indexJ, toSend, dataSock, udpAddress, udpPort);

}

else { // if the mode is set to "proverb" then send a proverb

System.out.println("Sending proverb to " + name);// prints to the console who will recieve the proverb

sendProverb(proverbs, indexP, toSend, dataSock, udpAddress, udpPort);

}

} catch (IOException x) { // catches an IO exception and prints an error message

System.out.println("Server read error");

x.printStackTrace();

} catch (InterruptedException ie) {ie.printStackTrace();

}

sock.close(); // closes the connection to this specific client

} catch (IOException ioe) {System.out.println(ioe);} // catches an IO exception and prints an error message

}

// the list builders for each Jokes and Proverbs. Here we take the name from the client and add it to the joke output.

static void jokeBuilder(String name, ArrayList<String> jokes){

jokes.add("JA " + name + ": I hope Elon Musk never gets involved in a scandal. Elongate would be really drawn out.");

jokes.add("JB " + name + ": My girlfriend is like the square root of -100. A solid 10 but also imaginary.");

jokes.add("JC " + name + ": I invited my girlfriend to go to the gym with me and then I didn\'t show. I hope she gets the message that we\'re not working out.");

jokes.add("JD " + name + ": As I handed my dad his 50th birthday card he looked at me with tears in his eyes and said\" Y'know one would have been enough.\"");

}

static void proverbBuilder(String name, ArrayList<String> proverbs){

proverbs.add("PA " + name + ": He who learns but does not think is lost! He who thinks but does not learn is in great danger. -Confucius");

proverbs.add("PB " + name + ": Wise men speak because they have something to say; Fools because they have to say something. -Plato");

proverbs.add("PC " + name + ": An eye for an eye only ends up making the whole world blind. -Mahatma Gandhi");

proverbs.add("PD " + name + ": Science my lad is made up of mistakes but they are mistakes which it is useful to make because they lead little by little to the truth. -Jules Verne");

}

// the calls to send the joke or proverb to the UDP server. The Client will connect to the UDP server to collect the joke/proverb

static void sendJoke(ArrayList<String> list, int jDex, StringBuilder sb, DatagramSocket ds, InetAddress udpAddress, int udpPort) {

byte[] packet = new byte[1024];

String jp = list.get(jDex); // grabs the randomly selected joke from the list

sb.append(jp + ", " + jDex + ", " + 5 + ", \n"); // appends the joke, used index number, dummy index number, and a line break to the StringBuilder to be sent to the client

packet = sb.toString().getBytes(); // converts the StringBuilder to a string, then to bytes

DatagramPacket dataPack = new DatagramPacket(packet, packet.length, udpAddress, udpPort); // creates a packet from the converted bytes

try {

ds.send(dataPack); // sends the packet to the UDP Server

} catch (IOException ioe) {ioe.printStackTrace(); //catches IO exceptions and prints an error message

}

ds.close(); // closes the connection to the UDP server

}

static void sendProverb(ArrayList<String> list, int pDex, StringBuilder sb, DatagramSocket ds, InetAddress udpAddress, int udpPort) {

byte[] packet = new byte[1024];

String jp = list.get(pDex); // grabs the randomly selected joke from the list

sb.append(jp + ", " + 5 + ", " + pDex + ", \n"); // appends the proverb, dummy index number, used index number, and a line break to the StringBuilder to be sent to the client

packet = sb.toString().getBytes(); // converts the StringBuilder to a string, then to bytes

DatagramPacket dataPack = new DatagramPacket(packet, packet.length, udpAddress, udpPort); // creates the packet from the converted bytes

try {

ds.send(dataPack); // sends the packet to the UDP Server

} catch (IOException ioe) {ioe.printStackTrace(); //catches IO exceptions and prints an error message

}

ds.close(); // closes the connection to the UDP server

}

}

// Communicates with the Admin Client to set the mode or shut the server down. Input from

// the admin is a single character. Admin controls are posted in the Admin Client console.

class AdminWorker extends Thread {

Socket sock;

AdminWorker (Socket s) {sock = s;}

public static String mode = "joke";

public void run() { // runs the Admin server functions (mode swap or shut down)

PrintStream out = null;

BufferedReader in = null;

try { // tries to create an input reader and an output reader

in = new BufferedReader(new InputStreamReader (sock.getInputStream()));

out = new PrintStream(sock.getOutputStream());

try { // takes the input from the Admin Client, processes the command, and will do 1 of 3 things

// 1) switch the mode to Joke and print a message confirming the switch, 2) same as 1), but with Proverb, 3) Shut the server down

// 4) do nothing and post a message on the Server.

String modeInput;

modeInput = in.readLine(); // takes the input

switch (modeInput) {

case "j":

modeSwitch("j");

break;

case "p":

modeSwitch("p");

break;

case "s":

System.out.println("Shutting down due to Admin command.");

System.exit(1);

break;

default:

System.out.println("Invalid command.");

break;

}

} catch (IOException x) { // catches an IO exception and prints an error message

System.out.println("Server read error");

x.printStackTrace();

}

sock.close(); // closes the connection to this specific client

} catch (IOException ioe) {System.out.println(ioe);} // catches an IO exception and prints an error message

}

// Switches the mode based on the input from the Admin Client

static void modeSwitch(String m){

if (m == "j"){

mode = "joke";

System.out.println("Currently telling jokes.");

}

else if ( m == "p"){

mode = "proverb";

System.out.println("Currently telling proverbs.");

}

}

}

// Creates the loop that connects to the Admin Client. The server will wait and listen for the

// Admin to send a command. Neither Admin Client nor JokeServer is needed to be active when starting

// the other.

class AdminLooper implements Runnable {

public static boolean adminControlSwitch = true;

public void run(){

System.out.println("In the admin looper thread");

int q\_len = 6;

int port = 4555; // the Admin Client's port, different fromt the Client port

Socket sock;

try{

ServerSocket servsock = new ServerSocket(port, q\_len);

while (adminControlSwitch) {

sock = servsock.accept();

new AdminWorker (sock).start();

}

}catch (IOException ioe) {System.out.println(ioe);}

}

}

public class AsyncJokeServer{

//Starts the server and deploys workers as needed when each Client connects

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

int q\_len = 6;

int port = 4545; // the port connection the server is listening to for Clients

Socket sock;

// Initializes the loop for the Admin Client to connect to

AdminLooper AL = new AdminLooper();

Thread t = new Thread(AL);

t.start();

ServerSocket servsock = new ServerSocket(port, q\_len); //initializes the server socket and port

System.out.println ("Vivek Patel's AsyncJokeServer 1.0 starting up, listening at port 4545 (client) and 4555 (admin).\n");

while (true) { // listens for a client to connect

sock = servsock.accept(); // accepts the client

new ClientWorker(sock).start(); // creates a worker for the client

}

}

}

import java.io.\*;

import java.net.\*;

import java.lang.\*;

import java.util.\*;

class UDPWorker extends Thread{ // Worker thread that will access the UDP server and recieve input from the Async Joke Server

int udpPort;

UDPWorker(int p){ // Constructor for UDP Worker

udpPort = p;

}

public void run(){ // runs the Worker thread

try{

DatagramSocket dataSock = new DatagramSocket(udpPort); // creats the connection to the UDP server

byte[] serverData = new byte[1024]; // creates a byte[] object to store the input from the UDP server

while(true){

DatagramPacket dataPack = new DatagramPacket(serverData, serverData.length); // creates a packet to be used to collect the packet from the UDP server

dataSock.receive(dataPack); // collects the packet from the UDP server

String data = new String(dataPack.getData()); // converts the packet to a string

String[] text = data.split(", "); // converts the string to a string list, separating the different values sent by the AsyncJokeServer

AsyncJokeClient.message = text[0]; // extracts the joke/proverb

// extracts the joke and proverb index numbers and converts them to ints.

String jTemp = text[1];

AsyncJokeClient.jDex = Integer.parseInt(jTemp);

String pTemp = text[2];

AsyncJokeClient.pDex = Integer.parseInt(pTemp);

AsyncJokeClient.ping = true; // lets the Client know that the message has arrived from the UDP server

}

}

catch(Exception e) { e.printStackTrace(); }

}

}

public class AsyncJokeClient {

static String message; // initializes the message string so the UDP worker can send it to the Client

static int jDex; // initializes the index numbers so the UDP worker can send it to the Client

static int pDex;

static boolean ping = false; // initializes the ping for the UDP worker to communicate with the Client on the status of the message

public static void main(String[] args) {

String serverName;

if (args.length < 1) serverName = "localhost"; //default server

else serverName = args[0]; // directed server input from the terminal

int udpPort = 9876; // Port for the UDP server

// initializes the index lists for both the joke and proverb states.

// builds the index lists from 0-3 as these are how many jokes/proverbs are in each list

// randomizes the indexes to create a random output

List<Integer> indexListJ = new ArrayList<>(4);

listBuilder(indexListJ);

Collections.shuffle(indexListJ);

List<Integer> indexListP = new ArrayList<>(4);

listBuilder(indexListP);

Collections.shuffle(indexListP);

System.out.println("Vivek Patel's AsyncJokeClient, 1.0.\n"); // prints to the console of the Client startup and server connection

System.out.println("Using server: " + serverName + ", Port: 4545");

BufferedReader in = new BufferedReader(new InputStreamReader(System.in)); // creates inputstream for the Client

try {

//initializes the variables necessary for the Client.

String name; // name provided to the Server by the user

String input; // the input stream from the user

System.out.print("What is your name?: "); // Asks the user for their name

System.out.flush(); // writes the data from the input stream from the user

name = in.readLine(); // saves the user's name

UDPWorker udpW = new UDPWorker(udpPort); // creates a new UDP worker at the referenced port number

udpW.start(); // starts the UDP worker

do {

// the user is prompted to press enter to recieve a joke or proverb. They can input "quit" to close the client.

// most random inputs are ignored. The only issue is if there "quit" anywhere in the input

System.out.print("Press <enter> for a joke or proverb or input (quit) to end: ");

System.out.flush();

input = in.readLine(); //writes the data from the input stream from the user

if (input.indexOf("quit") < 0) { // if "quit" isn't inputted, collect the necessary variables and call getJP

//to prepare the data to be sent to the server

getJP(name, indexListJ, indexListP, serverName, udpPort); // sends the request for a joke or proverb to the Server along with randomized index numbers

while (ping == false){ // while the UDP worker has not recieved a message, ask for 2 numbers to add together

System.out.print("While we wait, please enter 2 numbers separated by a space.: ");

input = in.readLine();

getSum(input); // takes the user input and sums the numbers together

}

System.out.println(message); // print the message when it arrives. The UDP worker will flag ping as true.

indexUpdate(jDex, pDex, indexListJ, indexListP); // update the index of the used joke/proverb

} ping = false; // reset the ping flag

} while (input.indexOf("quit") < 0); //if "quit" is anywhere within the input, close the client, otherwise keep looping the above code.

System.out.println("Cancelled by user request."); // This message is printed when the user "quit"s the client

System.exit(1); // kill the server

} catch (IOException ioe) {ioe.printStackTrace();} // catches IO exceptions and prints an error message to the client

}

// Creates the request and sends it to the AsyncJokeServer

private static void getJP(String name, List<Integer> indexListJ, List<Integer> indexListP, String serverName, int udpPort) {

Socket sock;

PrintStream toServer;

try {

sock = new Socket(serverName, 4545); // connects to the server at port 4545

toServer = new PrintStream(sock.getOutputStream()); //stream sends output to the server

// convert the index numbers and udp port number to strings

String indexJSend = indexListJ.get(0).toString();

String indexPSend = indexListP.get(0).toString();

String udpPortSend = Integer.toString(udpPort);

// sends the request to the AsyncJokeServer. The server will recieve the client name, index numbers for the joke and proverb lists, and the UDP port number

toServer.println(name + ", " + indexJSend + ", " + indexPSend + ", " + udpPortSend);

toServer.flush();

sock.close(); // closes the connection

} catch (IOException x) { // catches IO exceptions and prints an error message

System.out.println("Socket error.");

x.printStackTrace();

}

}

public static void indexUpdate(int j, int p, List<Integer> indexListJ, List<Integer> indexListP) { // updates the index that was used in the AsyncJokeServer

// determines if a joke or proverb was sent by comparing the two index numbers sent back.

// the used number will be between 0-3, so a dummy number of 5 is used to represent the unused index.

// takes the smallest value to determine the correct index list to update, then deletes the number from the index.

// since the client sends only the first number of a randomized list, the first number of the used index is deleted

// if the list is empty, send a message to the console and repopulate and randomize the list

if (jDex < pDex) {

indexListJ.remove(0);

if (indexListJ.size() == 0){

System.out.println("JOKE CYCLE COMPLETE");

listBuilder(indexListJ);

Collections.shuffle(indexListJ);

}

}

else {

indexListP.remove(0);

if (indexListP.size() == 0){

System.out.println("PROVERB CYCLE COMPLETE");

listBuilder(indexListP);

Collections.shuffle(indexListP);

}

}

}

private static void listBuilder (List<Integer> list){ // tool to repopulate a list

list.add(0);

list.add(1);

list.add(2);

list.add(3);

}

private static void getSum (String ns){ // sums the two numbers from a user input

String[] toSum = ns.split(" "); // converts the input string to an array

String a = toSum[0]; // pulls the two numbers from the array

String b = toSum[1];

int x = Integer.parseInt(a); // converts the strings to int

int y = Integer.parseInt(b);

int sum = x + y; // sums the two ints

System.out.println("The sum of " + a + " + " + b + " is: " + sum + ("!")); // print the completes sum statement

}

}

import java.io.\*;

import java.net.\*;

public class AsyncJokeAdminClient {

public static void main(String[] args) { // runs the Admin Client

String serverName;

if (args.length < 1) serverName = "localhost"; //default server

else serverName = args[0]; // directed server input from the terminal

System.out.println("Vivek Patel's AsyncJokeAdminClient, 1.0.\n");

System.out.println("Using server: " + serverName + ", Port: 4555");

BufferedReader in = new BufferedReader(new InputStreamReader(System.in)); // creates inputstream for the Admin

try {

String mode; // creates the mode variable for the Admin. This keeps the input from the Admin.

do {

// The Admin Client's console will print the list of commands for the Admin. It can change the mode based on the input,

// shut down the server, or exit the Admin Client

System.out.print("Input (j) for joke or (p) for proverb, (quit) to quit the Admin Client, or (s) to shut down the server: ");

System.out.flush(); //writes the data from the input stream from the user

mode = in.readLine(); // saves the Admin's input

if (mode.indexOf("quit") < 0) // if "quit" isn't inputted, send the Admin's command to the server

adminControl(mode, serverName);

} while (mode.indexOf("quit") < 0); //if "quit" is anywhere within the input, close the client.

System.out.println("Cancelled by user request."); // printed when the Admin closes the client.

} catch (IOException x) {x.printStackTrace();}

}

static void adminControl(String mode, String serverName) { //contact's the server with the Admin's command

Socket sock;

BufferedReader fromServer;

PrintStream toServer;

String textFromServer;

try {

sock = new Socket(serverName, 4555); // connects to the server, uses a different port than regular Clients

fromServer = new BufferedReader(new InputStreamReader(sock.getInputStream())); //stream recieves input from the server

toServer = new PrintStream(sock.getOutputStream()); //stream sends output to the server

toServer.println(mode); toServer.flush(); // sends the data and makes sure of it

sock.close(); // closes the connection to the server

} catch (IOException x) { // catches an IO exception and prints it

System.out.println("Socket error.");

x.printStackTrace();

}

}

}

/\*

I show here that I am able to continously type in 2 numbers to be summed together while waiting for the 40 second sleep timer. Most other functionality from the JokeServer assignment is the same, such as the random joke/proverb order and the state of the client.

\*/

//AsyncJokeServer:

In the admin looper thread

Vivek Patel's AsyncJokeServer 1.0 starting up, listening at port 4545 (client) and 4555 (admin).

Sending joke to Vivek

Sending joke to Vivek

Currently telling proverbs.

Sending proverb to Vivek

Sending proverb to Vivek

Shutting down due to Admin command.

//AsyncJokeClient:

Vivek Patel's AsyncJokeClient, 1.0.

Using server: localhost, Port: 4545

What is your name?: Vivek

Press <enter> for a joke or proverb or input (quit) to end: 5 6

While we wait, please enter 2 numbers separated by a space.: 23 65

The sum of 23 + 65 is: 88!

While we wait, please enter 2 numbers separated by a space.: 252 653

The sum of 252 + 653 is: 905!

While we wait, please enter 2 numbers separated by a space.: 1542 6589

The sum of 1542 + 6589 is: 8131!

While we wait, please enter 2 numbers separated by a space.: 32166 23156

The sum of 32166 + 23156 is: 55322!

While we wait, please enter 2 numbers separated by a space.: 12 5

The sum of 12 + 5 is: 17!

JB Vivek: My girlfriend is like the square root of -100. A solid 10 but also imaginary.

Press <enter> for a joke or proverb or input (quit) to end:

While we wait, please enter 2 numbers separated by a space.: 1456 125

The sum of 1456 + 125 is: 1581!

While we wait, please enter 2 numbers separated by a space.: 65 152

The sum of 65 + 152 is: 217!

JC Vivek: I invited my girlfriend to go to the gym with me and then I didn't show. I hope she gets the message that we're not working out.

Press <enter> for a joke or proverb or input (quit) to end:

While we wait, please enter 2 numbers separated by a space.: 456 4568

The sum of 456 + 4568 is: 5024!

While we wait, please enter 2 numbers separated by a space.: 1236 54

The sum of 1236 + 54 is: 1290!

While we wait, please enter 2 numbers separated by a space.: 158 812

The sum of 158 + 812 is: 970!

While we wait, please enter 2 numbers separated by a space.: 156 352

The sum of 156 + 352 is: 508!

PC Vivek: An eye for an eye only ends up making the whole world blind. -Mahatma Gandhi

Press <enter> for a joke or proverb or input (quit) to end:

While we wait, please enter 2 numbers separated by a space.: 1531 1561

The sum of 1531 + 1561 is: 3092!

While we wait, please enter 2 numbers separated by a space.: 12365 12

The sum of 12365 + 12 is: 12377!

PD Vivek: Science my lad is made up of mistakes but they are mistakes which it is useful to make because they lead little by little to the truth. -Jules Verne

Press <enter> for a joke or proverb or input (quit) to end: quit

Cancelled by user request.

//AsyncJokeAdminClient:

Vivek Patel's AsyncJokeAdminClient, 1.0.

Using server: localhost, Port: 4555

Input (j) for joke or (p) for proverb, (quit) to quit the Admin Client, or (s) to shut down the server: p

Input (j) for joke or (p) for proverb, (quit) to quit the Admin Client, or (s) to shut down the server: s

Input (j) for joke or (p) for proverb, (quit) to quit the Admin Client, or (s) to shut down the server: quit

Cancelled by user request.