**B9-Write a program using TCP sockets for wired network to implement**

**a. Peer to Peer Chat b. Multiuser Chat**

**a. Peer to Peer Chat:**

**#!/usr/bin/env python ^M**

**^M**

**""" ^M**

**A simple echo server ^M**

**""" ^M**

**import socket ^M**

**from threading import Thread, Lock^M**

**import sys^M**

**def receiver(sock):^M**

**global flag^M**

**while flag:^M**

**data=sock.recv(size) #receive client data^M**

**'''Receive data from client''' #Receive client data^M**

**if data == 'quit':^M**

**sys.exit(0)^M**

**print "\t\t"+data ^M**

**flag = True^M**

**print "Thread Exiting"^M**

**^M**

**host = '' #Server IP addres^M**

**port = 50004 #Port address ^M**

**backlog = 5 #For binding connections^M**

**size = 1024 #Size of packet data to receive^M**

**flag = True^M**

**tcp p2p client code:**

**#!/usr/bin/env python ^M**

**^M**

**""" ^M**

**A simple peer to peer chat^M**

**lines: 14^M**

**""" ^M**

**^M**

**import socket ^M**

**from threading import Thread, Lock^M**

**import sys^M**

**^M**

**def receiver(sock):^M**

**while flag:^M**

**data = sock.recv(1024) #Receive data from server^M**

**if data == 'quit':^M**

**sys.exit(0)^M**

**print "\t\t"+data^M**

**^M**

**host = '172.168.255.121' #Server IP addres^M**

**port = 50004 #Port address ^M**

**size = 1024 #Size of packet data to receive^M**

**flag = True^M**

**^M**

**sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) #Creating a Socket^M**

**sock.connect((host,port)) #Connect to Server^M**

**^M**

**# Start Receiver Thread^M**

**hop = Thread(target=receiver, args=(sock,))^M**

**hop.daemon = True^M**

**hop.start() ^M**

**^M**

**data = "random" #Entering any value so it is not quit initially ^M**

**^M**

**while data != 'quit': #User will enter 'quit' to exit server^M**

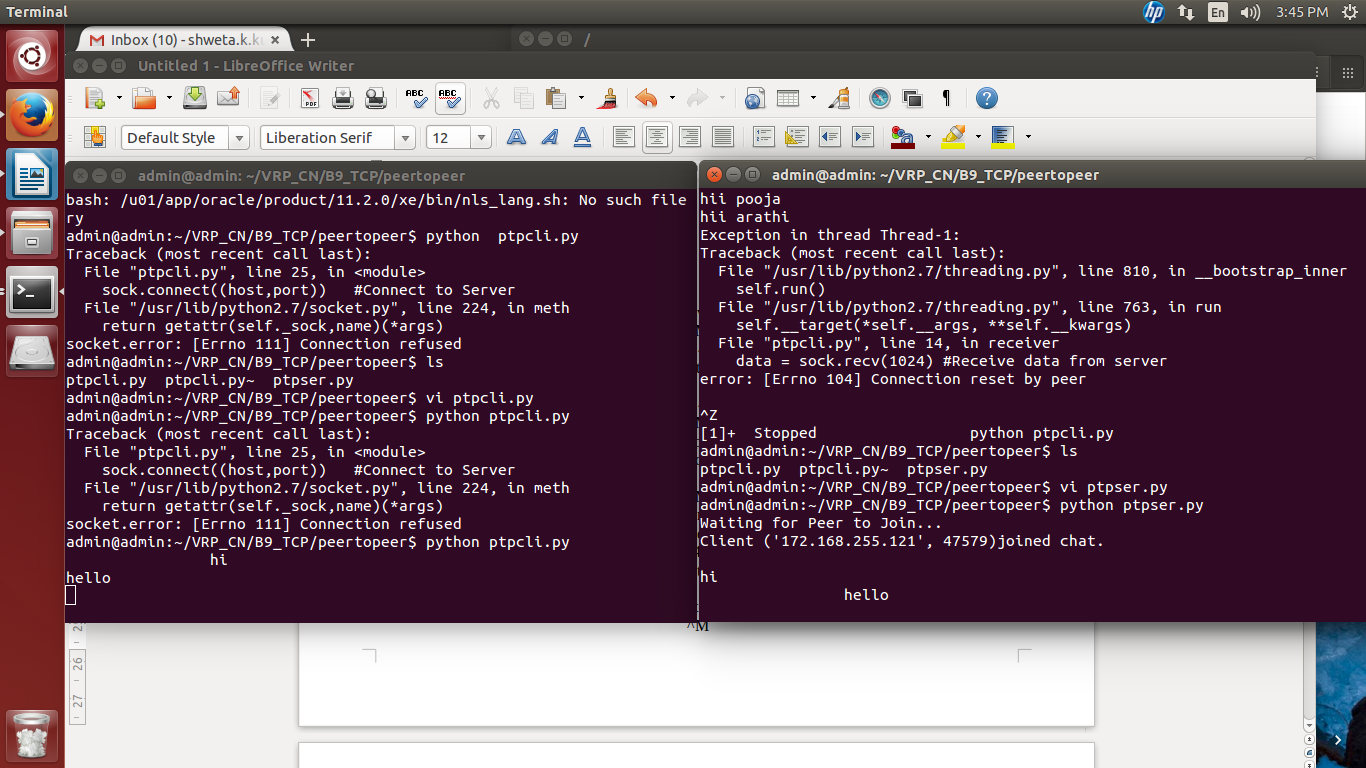
**data = raw\_input()#Enter message to send to server^M**

**sock.send(data) ^M**

**'''Send data to server''' #Sending data to server^M**

**sock.close() #Closing the socket^M**

**OUTPUT:**



**b. Multiuser Chat**

**^M**

**""" ^M**

**A simple TCP multiuser server ^M**

**""" ^M**

**^M**

**import socket^M**

**import os^M**

**from threading import Thread^M**

**import threading^M**

**import thread^M**

**^M**

**clients = set() #add data like list^M**

**clients\_lock = threading.Lock()^M**

**names = {} #like dic ^M**

**^M**

**def listener(client, address):^M**

**name=client.recv(1024) #Receive name of connected client^M**

**names[client] = name #Add name to names dictionary^M**

**print names[client]+" joined chat from IP:"+str(address[0])+" and Port:"+str(address[1])^M**

**with clients\_lock:^M**

**clients.add(client) #Adding client socket value to clients list^M**

**#print clients^M**

**try: ^M**

**while True:^M**

**data=client.recv(1024) #Receive data from client^M**

**if not data:^M**

**break^M**

**else:^M**

**data = names[client]+":"+str(data) #Append data with client name^M**

**#print data^M**

**with clients\_lock:^M**

**for c in clients: #Send the data to all the clients^M**

**c.send(data)^M**

**finally:^M**

**with clients\_lock:^M**

**clients.remove(client) #If client exit, then remove the client value^M**

**client.close()^M**

**^M**

**host = '172.168.255.121' #Server IP addres^M**

**port = 10015 #Port address ^M**

**"ser\_mtcp.py" 58 lines, 1837 characters**

**TCP MUTLICLIENT CODE:**

**#!/usr/bin/env python ^M**

**^M**

**import socket ^M**

**from threading import Thread, Lock^M**

**^M**

**def receiver(sock):^M**

**while flag:^M**

**data = sock.recv(size) ^M**

**print "\t\t"+data^M**

**^M**

**host = '172.168.255.121' #Server IP addres^M**

**port = 10015 #Port address ^M**

**size = 1024 #Size of packet data to receive^M**

**flag = True^M**

**^M**

**sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)**

**#Creating a Socket^M**

**sock.connect((host,port)) #Creating a Socket'''^M**

**^M**

**'''Connect to Server'''^M**

**^M**

**^M**

**# Start Receiver Thread^M**

**hop = Thread(target=receiver, args=(sock,))^M**

**hop.daemon = True^M**

**hop.start() ^M**

**^M**

**data = raw\_input("Enter Your name:") ^M**

**sock.send(data) ^M**

**while data != 'quit': #User will enter 'quit' to exit server^M**

**data = raw\_input() #Enter message to send to server^M**

**sock.send(data) #Sending data to server^M**

**^M**

**sock.close() #Closing the socket^M**

**OUTPUT:**

