|  |  |
| --- | --- |
| **Name** | Manish Shashikant Jadhav |
| **UID** | 2023301005 |
| **Subject** | Computer Communication and Networks (CCN) |
| **Experiment No.** | 2 |
| **Aim** | Network Socket Programming |
| **Code for Part 1:** | **Problem Statement:**  Implement the following rudimentary string processing application using connection oriented client-server programming. Some guidelines for the implementation are as follows. The client will send a textual paragraph terminated by ‘\n’ to the server (assume that in the paragraph, ‘.’ appears only at the end of sentences and nowhere else). The server will compute the number of characters, number of words, and number of sentences in the paragraph, and send these numbers back to the client. The client will print these numbers on the screen.   1. **Server:**   import socket  HOST = '127.0.0.1'  # Standard loopback interface address (localhost)  PORT = 65432        # Port to listen on (non-privileged ports are > 1023)  with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:      s.bind((HOST, PORT))      s.listen()      conn, addr = s.accept()      with conn:          print('Connected by', addr)          while True:              data = conn.recv(1024)              if not data:                  break              paragraph = data.decode('utf-8')              num\_chars = len(paragraph)              num\_words = len(paragraph.split())              num\_sentences = paragraph.count('.')              results = f"Characters: {num\_chars}\nWords: {num\_words}\nSentences: {num\_sentences}"              conn.sendall(results.encode('utf-8'))   1. **Client Side:**   import socket  HOST = '127.0.0.1'  # The server's hostname or IP address  PORT = 65432        # The port used by the server  with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:      s.connect((HOST, PORT))      paragraph = input("Enter a paragraph: ") + '\n'      s.sendall(paragraph.encode('utf-8'))      results = s.recv(1024).decode('utf-8')      print(results) |
| **Output:** |  |
| **Code for Part 2:** | **Problem Statement:**  Make it concurrent so that it can serve multiple clients at a time. (Multiple clients on multiple terminals and single server terminals).  **Server(Updated):**  import socket  import threading  HOST = '127.0.0.1'  PORT = 65432  def handle\_client(conn, addr):      print(f"Connected by {addr}")      while True:          data = conn.recv(1024).decode('utf-8')          if not data:              break          paragraph = data          num\_chars = len(paragraph)          num\_words = len(paragraph.split())          num\_sentences = paragraph.count('.')          results = f"Characters: {num\_chars}\nWords: {num\_words}\nSentences: {num\_sentences}"          conn.sendall(results.encode('utf-8'))      conn.close()  def start\_server():      with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:          s.bind((HOST, PORT))          s.listen()          print(f"Server listening on port {PORT}")          while True:              conn, addr = s.accept()              thread = threading.Thread(target=handle\_client, args=(conn, addr))              thread.start()  if \_\_name\_\_ == '\_\_main\_\_':      start\_server() |
| **Output:** | **Server (connected to 3 Clients):**    **Client 1:**    **Client 2:**    **Client 3:** |
| **Code for Part 3:** | 1. **Server Side:**   import socket  import threading  HOST = '127.0.0.1'  # Standard loopback interface address (localhost)  PORT = 65432        # Port to listen on  clients = []  client\_names = {}  def handle\_client(conn, addr, name):      global clients, client\_names      print(f'{name} connected from {addr}')      broadcast(f'{name} has joined the chat!', name)      while True:          try:              data = conn.recv(1024).decode()              if not data:                  break              message = f'{name}: {data}'              broadcast(message, name)          except ConnectionError:              print(f'{name} disconnected unexpectedly')              broadcast(f'{name} has left the chat!', name)              clients.remove(conn)              del client\_names[conn]              break      conn.close()  def broadcast(message, sender):      for client in clients:          if client != sender:              try:                  client.sendall(message.encode())              except ConnectionError:                  print(f'Error sending message to {client\_names[client]}')  with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:      s.bind((HOST, PORT))      s.listen()      print(f'Server listening on {HOST}:{PORT}')      while True:          conn, addr = s.accept()          name = conn.recv(1024).decode()          client\_names[conn] = name          clients.append(conn)          client\_thread = threading.Thread(target=handle\_client, args=(conn, addr, name))          client\_thread.start()   1. **Client Side:**   import socket  HOST = '127.0.0.1'  # The server's hostname or IP address  PORT = 65432        # The port used by the server  with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:      s.connect((HOST, PORT))      name = input("Enter your name: ")      s.sendall(name.encode())      while True:          message = input()          if message == '/quit':              break          s.sendall(message.encode())          data = s.recv(1024).decode()          print(data) |
| **Output:** | 1. **Server:**      1. **Client 2:**      1. **Client 3:** |
| **Conclusion** | Hence by completing this experiment I came to know about Network Socket Programming |