```
_abhishek@hp in repo: CN/05-Lab on ₽ main [!+] via ♣ v3.11.3 took 4ms
                                                                                —abhishek@hp in repo: CN/05-Lab on 毕 main [!+] via ♣ v3.11.3 took 4ms
 python server.py

    □) python client.py

[SERVER] Keep the files to send/recieve inside folder: 'server/'
                                                                               [CLIENT] Keep the files to send/recieve inside folder: 'client/'
[STARTING] Server is listening on :53535
                                                                               [CONNECTED] Client connected to 127.0.0.1:53535
[NEW CONNECTION] 127.0.0.1:54832 connected.
                                                                               (file_name)> file.txt
[ACTIVE CONNECTIONS] 1
                                                                               [SERVER] File name recieved
[127.0.0.1:54832] Create file 'file.txt'
                                                                               [SERVER] File contents of 'file.txt' recieved
[127.0.0.1:54832] Contents of 'file.txt': hello world from client:)
                                                                               [SERVER] Create file 'file_s.txt'
                                                                               [SERVER] Contents of 'file_s.txt': hello world from server :)
(ip:port)> 127.0.0.1:54832
(file_name)> file_s.txt
                                                                               (file_name)> QUIT!
[127.0.0.1:54832] File name recieved
[127.0.0.1:54832] File contents of 'file_s.txt' recieved
                                                                               [CLIENT] Client is shutting down...
[DISCONNECTED] 127.0.0.1:54832 disconnected.
[ACTIVE CONNECTIONS] 0
                                                                                —abhishek@hp in repo: CN/05-Lab on բ main [!+] via ಈ v3.11.3 took 40s
(ip:port)> ☐
```

05-Lab > 🥏 server.py > ...

```
import socket
     import threading
     import os
     import time
     import readline
     from urllib.parse import quote, unquote # for encoding/decoding data in url format
     from dataclasses import dataclass
     # IP = socket.gethostbyname(socket.gethostname())
     IP = ''
10
     PORT = 53535
11
     ADDR = (IP, PORT)
     SIZE = 1024
    FORMAT = "utf-8"
14
     DISCONNECT_MESSAGE = "QUIT!"
15
16
    # MSG Format:
17
    # MSG/<msg>
18
    # CREATE/<file_name>
19
    # POST/<file_name>/<file_content>
20
21
22
     folder_path = "server/" # folder to store files for server
23
24
     clients = [] # list of all clients connected to server
25
26
     # Client class to store client connection and address
27
     @dataclass
28
     class Client:
29
30
         conn: socket.socket
31
         addr: str
32
33
    # find client by address
34
     def find_client(addr):
35
         for client in clients:
36
37
             if client.addr = \alpha ddr:
                 return client
38
         return None
39
```

```
42
     current_prompt = "" # store current prompt of input
43
     def print_msg(msg):
44
         if not current_prompt:
             print(msg)
46
47
         else:
             input_buffer = readline.get_line_buffer() # store current input buffer
48
             print(f"\r{msg}\n{current_prompt}{input_buffer}", end="", flush=True) # print message and restore input buffer
49
50
51
52
     def input_msg(string):
         global current_prompt
53
54
         current_prompt = string
56
         result = input(f"\r{string}")
57
58
         current_prompt = ""
59
60
         return result
61
62
    # send message to client
63
     def send_msg(conn, msg):
64
         conn.send(f"MSG/{quote(msg)}".encode(FORMAT))
65
66
67
68
     # take input from server
69
     def server_input():
         while True:
70
             to_addr = input_msg("(ip:port)> ").strip()
71
             if to_addr = DISCONNECT_MESSAGE: # disconnect server
72
                 end_server()
73
             elif to_addr in ["", "list", "ls"]: # list all clients
74
                 print_msg("Active clients:")
75
76
                 for client in clients:
                     print_msg(f" {client.addr}")
77
                 continue
78
79
             to_client = find_client(to_addr) # find client by address
80
```

```
if to_client is None:
            print_msg(f"Error: Client '{to_addr}' not found.")
            continue
        msg = input_msg("(file_name)> ") # take file name
        if msg = DISCONNECT_MESSAGE:
            end_server()
        elif msg = "":
            continue
        # send file name to client
        to_client.conn.send(f"CREATE/{quote(msg)}".encode(FORMAT))
# handle the file requests from client
def handle_file_request(client, request, body):
    # CREATE/<file_name>
    if request = "CREATE":
        file_name = unquote(body) # decode file name
        file_path = folder_path + file_name
        print_msg(f"[{client.addr}] Create file '{file_name}'")
        with open(file_path, "w"): # create file
            msg = f"File name recieved"
            send_msg(client.conn, msg) # send acknowledgement to client
           time.sleep(0.1)
            client.conn.send(f"GET/{quote(file_name)}".encode(FORMAT)) # send GET request to client to get file contents
    elif request = "POST":
        file_name, file_content = map(unquote, body.split("/", 1)) # decode file name and content
        file_path = folder_path + file_name
        print_msg(f"[{client.addr}] Contents of '{file_name}': {file_content}")
        with open(file_path, "a") as f:
           f.write(file_content) # write file content to file
            msg = f"File contents of '{file_name}' recieved"
            send_msq(client.conn, msg) # send acknowledgement to client
    # GET/<file_name>
    elif request = "GET":
        file_name = unquote(body) # decode file name
```

83 84

85

86 87

88

89 90

91

96

97

98

99

100

101102

103

104

105

106 107 108

109

110 111

112

113

114

115

116 117 118

119

```
file_path = folder_path + file_name
        try:
            with open(file_path, "r") as f: # read file content
                file_content = f.read()
                client.conn.send(f"POST/{quote(file_name)}/{quote(file_content)}".encode(FORMAT)) # send file content to client
        except Exception as e: # if file not found
            err = f"Error: {e}"
            print_msg(err)
            send_msg(client.conn, err)
    else:
        send_msg(client.conn, f"Error: Invalid request '{request}'.")
# handle client connection
def handle_client(conn, addr):
    print_msg(f"[NEW CONNECTION] {addr} connected.")
    while True:
        msg = conn.recv(SIZE).decode(FORMAT) # recieve message from client
        if not msg or msg = DISCONNECT_MESSAGE:
            break
        try:
            request, body = msg.split("/", 1) # split request and body
        except Exception as e:
            err = "Error: Invalid request format."
            print_msg(f"[{addr}] {err}")
            send_msg(conn, err)
            continue
        if request = "MSG": # if message request
            print_msg(f"[{addr}] {unquote(body)}")
            continue
        try: # handle file request
            handle_file_request(find_client(addr), request, body)
        except Exception as e: # if any error while handling file request
            err = f"Error: {e}"
            print_msg(f"[{addr}] {err}")
```

123

124

125

126

127

128

129 130

131

132133134

135 136

137138

139

140

141142

143 144

145

146

147

148

149

150 151 152

153

154 155

156

157

158

159

```
161
                  send_msg(conn, err)
162
                  continue
163
          conn.close() # close connection
164
          clients.remove(Client(conn, addr)) # remove client from list
165
          print_msg(f"[DISCONNECTED] {addr} disconnected.")
166
          print_msg(f"[ACTIVE CONNECTIONS] {threading.active_count()-3}")
167
168
      def main():
169
          server = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # create server socket
170
          server.bind(ADDR) # bind server socket to address
171
          server.listen() # start listening for connections
172
173
          print_msg(f"[STARTING] Server is listening on {IP}:{PORT}")
174
          server_input_thread = threading.Thread(target=server_input)
175
          server_input_thread.start() # start server input thread
176
177
178
          while True:
              conn, addr = server.accept() # accept connection
179
180
              addr = f"{addr[0]}:{addr[1]}"
181
              clients.append(Client(conn, addr)) # add client to list
182
183
              thread = threading. Thread(target=handle_client, args=(conn, addr))
184
              thread.start() # start client thread
185
186
187
              print_msg(f"[ACTIVE CONNECTIONS] {threading.active_count()-2}")
188
189
      def end_server():
190
          print_msg("\n[EXITING] Server is shutting down...")
191
          for client in clients: # send disconnect message to all clients
192
              client.conn.send(DISCONNECT_MESSAGE.encode(FORMAT))
193
          os._exit(0)
194
195
196
      if __name__ = "__main__":
          if not os.path.exists(folder_path): # create folder if not exists
197
              os.makedirs(folder_path)
198
          print_msg(f"[SERVER] Keep the files to send/recieve inside folder: '{folder_path}'")
199
200
          try:
```

201	main()
202	except KeyboardInterrupt: # if server is stopped
203	end_server()

```
05-Lab > 🥏 client.py > ...
```

```
import socket
     import threading
     import os
     import time
     import readline
     from urllib.parse import quote, unquote # for encoding/decoding data in url format
     IP = socket.gethostbyname(socket.gethostname())
    # IP = "172.16.19.141"
    PORT = 53535
11
    # PORT = 8006
    ADDR = (IP, PORT)
    SIZE = 1024
14
    FORMAT = "utf-8"
15
    DISCONNECT_MESSAGE = "QUIT!"
16
17
     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # create client socket
18
     connected = True # flag to check if client is connected to server
19
20
21
    # MSG Format:
   # MSG/<msg>
22
   # CREATE/<file_name>
23
    # POST/<file_name>/<file_content>
24
25
26
     folder_path = "client/" # folder to store files for client
27
28
     current_prompt = "" # store current prompt of input
29
30
31
     def print_msg(msg):
         if not current_prompt:
32
             print(msg)
33
         else:
34
             input_buffer = readline.get_line_buffer() # store current input buffer
35
             print(f"\r{msg}\n{current_prompt}{input_buffer}", end="", flush=True) # print message and restore input buffer
36
37
38
     def input_msg(string):
39
         global current_prompt
40
```

```
41
42
         # set current prompt and take input
43
         current_prompt = string
         result = input(f"\r{string}")
44
         current_prompt = ""
46
47
         return result
48
49
50
    # send message to server
     def send_msg(conn, msg):
51
52
         conn.send(f"MSG/{quote(msg)}".encode(FORMAT))
53
54
55
    # handle file requests from server
     def handle_file_request(request, body):
56
         # CREATE/<file_name>
57
         if request = "CREATE":
58
             file_name = unquote(body) # decode file name
59
             file_path = folder_path + file_name
60
61
             print_msg(f"[SERVER] Create file '{file_name}'")
             with open(file_path, "w"): # create file
62
                 msg = f"File name recieved"
63
                 send_msg(client, msg) # send acknowledgement to server
64
65
                 time.sleep(0.1)
                 client.send(f"GET/{quote(file_name)}".encode(FORMAT)) # send GET request to server to get file contents
66
67
68
         # POST/<file_name>/<file_content>
         elif request = "POST":
69
70
             file_name, file_content = map(unquote, body.split("/", 1)) # decode file name and file content
             file_path = folder_path + file_name
71
             print_msg(f"[SERVER] Contents of '{file_name}': {file_content}")
72
             with open(file_path, "a") as f:
73
                 f.write(file_content) # write file content to file
74
                 msg = f"File contents of '{file_name}' recieved"
75
                 send_msg(client, msg) # send acknowledgement to server
76
77
         # GET/<file_name>
78
         elif request = "GET":
79
             file_name = unquote(body) # decode file name
80
```

```
file_path = folder_path + file_name
        try:
            with open(file_path, "r") as f: # read file contents
                file_content = f.read()
                client.send(f"POST/{quote(file_name)}/{quote(file_content)}".encode(FORMAT)) # send file contents to server
        except Exception as e: # if file not found
            err = f"Error: {e}"
            print_msg(err)
            send_msg(client, err)
    else:
        send_msg(client, f"Error: Invalid request '{request}'.")
def handle_server():
    global connected
    while connected:
        msg = client.recv(SIZE).decode(FORMAT) # recieve message from server
        if not msg or msg = DISCONNECT_MESSAGE:
            connected = False
            end_client()
        try:
            request, body = msg.split("/", 1) # split request and body
        except Exception as e: # if invalid request format
            err = "Error: Invalid request format."
            print_msg(err)
            send_msg(client, err)
            continue
        if request = "MSG": # if message request
            print_msg(f"[SERVER] {unquote(body)}")
            continue
        try: # handle file request
            handle_file_request(request, body)
        except Exception as e: # if error occured while handling file request
            err = f"Error: {e}"
            print_msg(err)
            send_msg(client, err)
```

82 83

84

87

88

89 90 91

92

93 94

95

96

97

98

99

100 101

102103

104

105

106

107108

109 110

111

112

113 114

115

116117

118

119

```
121
                  continue
122
          client.close() # close connection
123
          print_msg(f"[DISCONNECTED] Server disconnected from Client.")
124
          os._exit(0)
125
126
      def main():
127
          client.connect(ADDR) # connect to server
128
129
          print_msg(f"[CONNECTED] Client connected to {IP}:{PORT}")
130
131
132
          server_thread = threading.Thread(target=handle_server)
133
          server_thread.start() # start thread to handle server
134
          global connected
135
          while connected:
136
              msg = input_msg("(file_name)> ").strip() # take input from user
137
              if msg = DISCONNECT_MESSAGE:
138
                  end_client()
139
140
              client.send(f"CREATE/{quote(msg)}".encode(FORMAT)) # send CREATE request to server
141
142
          client.close() # close connection
143
          print_msg(f"[DISCONNECTED] Client disconnected from {IP}:{PORT}")
144
145
146
147
      def end_client():
          client.close() # close connection
148
          print_msg("\n[CLIENT] Client is shutting down...")
149
          os._exit(0)
150
151
      if __name__ = "__main__":
152
          if not os.path.exists(folder_path): # create folder if not exists
153
              os.makedirs(folder_path)
154
          print_msg(f"[CLIENT] Keep the files to send/recieve inside folder: '{folder_path}'")
155
156
          try:
              main()
157
          except KeyboardInterrupt: # if client is stopped
158
              end_client()
159
```