COMP 7005
Assignment 2

Design

Daryush Balsara
A01265967
Oct, 4 2024

Purpose

- To implement a client-server application using TCP over the network and gain experience in network communication, socket programming, file handling, and concurrency

Functions

client.py

parse_agrs()	Parses command-line arguments
connect_to_server(ip, port)	Creates a TCP socket and attempts to connect to the server using the specified IP address and port
send_file(sock, file_path)	Sends the specified file to the server through the established socket. It first sends the file path, followed by the file content in chunks of a defined size
receive_response(sock)	Receives and decodes a response from the server through the specified socket, returning the received data as a string.
main()	It parses arguments, establishes a connection to the server, sends the specified file, and receives the server's response regarding the number of alphabetic letters in the file.

server.py

parse_agrs()	Parses command-line arguments
setup_server_socket(PORT)	Creates a TCP socket, binds it to the specified port and host (0.0.0.0), and starts listening for incoming connections.
wait_for_connection(server_sock)	Waits for an incoming connection on the server socket. When a client connects, it returns the connection object and the client address.

count_alphabetical(file_path)	Reads the specified file and counts the number of alphabetic characters in its content.
handle_client(conn)	Manages the communication with a connected client. It receives the file path from the client, counts the alphabetic letters in the specified file, and sends the result back to the client.
send_reply(conn, reply)	Sends a reply message back to the connected client through the specified connection.
main()	Sets up the server. It parses arguments, creates the server socket, and enters a loop to accept client connections, spawning a new thread to handle each client

Variables

client.py

LINE_LEN	Set to 4096 and defines the maximum size, in bytes, for data chunks that will be sent or received over the TCP socket

server.py

LINE_LEN	Set to 4096 and defines the maximum size, in bytes, for data chunks that will be sent or received over the TCP socket
HOST	Set to "0.0.0.0" and specify that the server should listen for incoming connections and accept connections from any IP address

Pseudo Code

client.py

parse_args

Parameters

Parameter	Type	Description
na	na	na

Return

Value	Reason
parse_args	Namespace containing the parsed command-line arguments, allowing the rest of the program to access the file paths provided by the user

```
function parse_args():
    create parser with description "Client-server application using TCP
sockets over the network”
    add argument '-ip' or '--ip' to parser with type=ip_address,
required=True, and help message "Accepts the IP address to send on"
    add argument '-p' or '--port' to parser with type=int, required=True, and
help message "Accepts the port to send on"
    add argument '-f' or '--file' to parser with type=str, required=True, and
help message "Path of the file to send"
    return parser.parse_args()
```

send_file

Parameters

Parameter	Type	Description
sock	na	na

Return

Value	Reason
na	na

```
function send_file(sock, file_path)
    TRY
        IF file_path does not exist
            PRINT "Error: The file '{file_path}' does not exist."
            RETURN

        PRINT "Sending file path: {file_path}"
        SEND file_path encoded in 'utf-8' through the socket
        SEND a null byte through the socket

        PRINT "Sending file content of: {file_path}"
        OPEN file at file_path in binary read mode AS file
            WHILE true
                READ LINE_LEN bytes from the file into file_data
                IF file_data is empty
                    BREAK
                SEND file_data through the socket
        PRINT "File sent successfully."
    EXCEPT FileNotFoundError
        PRINT "Error: File '{file_path}' not found."
    EXCEPT PermissionError
        PRINT "Error: Permission denied to read '{file_path}'."
    EXCEPT Exception AS e
        PRINT "Error: Unable to send file: {e}"
```

reveive_response

Parameters

Parameter	Type	Description
na	na	na

Return

Value	Reason
parse_args	Namespace containing the parsed command-line arguments, allowing the rest of the program to access the file paths provided by the user

```
function receive_response(sock)
    TRY
```

```
        RECEIVE response from the socket with LINE_LEN bytes
        RETURN response decoded as 'utf-8'
    EXCEPT Exception AS e
        PRINT "Error: Unable to receive response: {e}"
        EXIT with status code 1

function main
    args = CALL parse_args()
    sock = CALL connect_to_server(args.ip, args.port)

    TRY
        CALL send_file(sock, args.file)
        response = CALL receive_response(sock)
        PRINT "Number of alphabetic letters: {response}"
    FINALLY
        IF sock exists
            CLOSE the socket
```

server.py

parse_args

Parameters

Parameter	Type	Description
na	na	na

Return

Value	Reason
parse_args	Namespace containing the parsed command-line arguments, allowing the rest of the program to access the file paths provided by the user

```
function parse_args():
    create parser with description "Client-server application using TCP
sockets over the network”
    add argument '-p' or '--port' to parser with type=int, required=True, and
help message "Accepts the port to listen on"
    return parser.parse_args()
```

setup_server_socket

Parameters

Parameter	Type	Description
PORT	int	The port number on which the server will listen.

Return

Value	Reason
server_socket	Returns the server socket

```
function setup_server_socket(PORT)
    CREATE connection tuple (HOST, PORT)
    TRY
        CREATE a TCP socket using IPv4 (AF_INET)
        BIND the socket to the connection
        LISTEN for incoming connections (maximum 5)
        PRINT "Server listening on port {PORT}"
        RETURN the server socket
    EXCEPT Exception AS e
        PRINT "Error: Unable to create server socket: {e}"
        EXIT with status code 1
```

wait_for_connection

Parameters

Parameter	Type	Description
server_sock	socket.socket	The server socket that is listening for connections.

Return

Value	Reason
conn	Returns the client connection socket

```
function wait_for_connection(server_sock)
    TRY
        ACCEPT an incoming connection from server_sock
        PRINT "Server is listening on {client_addr}"
        RETURN the connection and client address
    EXCEPT Exception AS e
        PRINT "Error: Unable to accept connection: {e}"
        RETURN None
```

count_alphabetical

Parameters

Parameter	Type	Description
file_path	str	The path to the file whose alphabetic characters need to be counted.

Return

Value	Reason
alphabetical_count	Returns the number of alphabetical letters

```
function count_alphabetical(file_path)
    TRY
        OPEN the file at file_path in binary read mode AS file
            READ the entire data from the file
            COUNT the number of alphabetic characters in the decoded data
            RETURN the alphabetic count
    EXCEPT Exception AS e
        PRINT "Error: Unable to count alphabetic letters: {e}"
        RETURN None
```

handle_client

Parameters

Parameter	Type	Description
conn	socket.socket	The client connection socket

Return

Value	Reason
na	na

```
function handle_client(conn)
    TRY
        INITIALIZE an empty bytearray file_path_bytes
        WHILE true
            RECEIVE a chunk of data from conn
            EXTEND file_path_bytes with the chunk
            IF null byte is in the chunk
                BREAK

        DECODE file_path_bytes into a string up to the null byte

        PRINT "Received file request: {file_path}"

        CALL count_alphabetical(file_path) to get the alphabetic count
        IF alphabetic_count is not None
            CALL send_reply(conn, str(alphabetic_count))
        ELSE
            CALL send_reply(conn, "Error: Unable to count characters in file
'{file_path}'")

    EXCEPT Exception AS e
        PRINT "Error handling request: {e}"
    FINALLY
        CLOSE the connection
```

send_reply

Parameters

Parameter	Type	Description
conn	socket.socket	The client connection socket
reply	str	The reply message to be sent to the client

Return

Value	Reason
alphabetical_count	Returns the number of alphabetical letters

```
function send_reply(conn, reply)
    TRY
        SEND the reply encoded in 'utf-8' through conn
    EXCEPT BrokenPipeError
        PRINT "Error: Client disconnected unexpectedly."
    EXCEPT Exception AS e
        PRINT "Error: Unable to send response: {e}"
```

main

Parameters

Parameter	Type	Description
na	na	na

Return

Value	Reason
na	na

```
function main()
    args = CALL parse_args()
    PORT = args.port

    CALL setup_server_socket(PORT) to get server_sock

    TRY
        WHILE true
            CALL wait_for_connection(server_sock) to get conn
            IF conn is not None
                CREATE a new thread for handle_client with conn as argument
                START the client thread
```

```
EXCEPT KeyboardInterrupt
    PRINT "\nShutting down and closing the connection"
FINALLY
    CLOSE the server socket
```