*Design*

[**Purpose** 4](#_Toc188772622)

[**Data Types** 4](#_Toc188772623)

[Arguments 4](#_Toc188772624)

[Settings 4](#_Toc188772625)

[**Functions** 5](#_Toc188772626)

[Client 5](#_Toc188772627)

[Server 6](#_Toc188772628)

[**States** 6](#_Toc188772629)

[Client 6](#_Toc188772630)

[Server 6](#_Toc188772631)

[**State Table** 7](#_Toc188772632)

[Client 7](#_Toc188772633)

[Server 7](#_Toc188772634)

[**State Transition Diagram** 8](#_Toc188772635)

[**Pseudocode** 9](#_Toc188772636)

[Client 9](#_Toc188772637)

[validate\_arguments: 9](#_Toc188772638)

[prepare\_filename: 9](#_Toc188772639)

[open\_file: 10](#_Toc188772640)

[get\_file\_size: 10](#_Toc188772641)

[read\_file\_content: 10](#_Toc188772642)

[close\_file: 11](#_Toc188772643)

[create\_client\_socket: 11](#_Toc188772644)

[connect\_to\_server: 12](#_Toc188772645)

[send\_message\_to\_server: 13](#_Toc188772646)

[receive\_server\_response: 13](#_Toc188772647)

[close\_socket: 14](#_Toc188772648)

[Server 15](#_Toc188772649)

[validate\_arguments: 15](#_Toc188772650)

[setup\_server\_socket: 15](#_Toc188772651)

[accept\_client\_connections: 16](#_Toc188772652)

[process\_client\_message: 17](#_Toc188772653)

[caesar\_encrypt: 18](#_Toc188772654)

[handle\_signal: 19](#_Toc188772655)

[cleanup: 19](#_Toc188772656)

# **Purpose**

The programs implement inter-process communication using UNIX domain sockets. The server encrypts messages using the Caesar cipher with a shift provided at runtime, while the client reads a file, sends its content to the server, and displays the encrypted message received in response.

Both server and client program accept one command line argument as follows:

* ./server <Shift Value in Integer>
* ./client <file location>

# **Data Types**

## Arguments

### Client

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| argv | string[] | arguments |
| argc | integer | number of arguments |
| program\_name | string | name of the program |
| file name | string | name of the file to encrypt |

### Server

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| argv | string[] | arguments |
| argc | integer | number of arguments |
| program\_name | string | name of the program |
| shift value | integer | value to shift alphabets by |

## Settings

### Client

|  |  |  |
| --- | --- | --- |
| **Field** | **Value** | **Description** |
| S\_PATH | string | Path for the UNIX domain socket file. |
| FILE\_NAME\_SIZE | integer | Maximum length of file name. |
| BUFFER\_SIZE | integer | Size of the buffer for communication. |

### Server

|  |  |  |
| --- | --- | --- |
| **Field** | **Value** | **Description** |
| S\_PATH | string | Path for the UNIX domain socket file. |
| BACKLOG | integer | Maximum number of pending connections. |
| BUFFER\_SIZE | integer | Size of the buffer for communication. |

### Context Client

|  |  |  |
| --- | --- | --- |
| **Field** | **Value** | **Description** |
| File | string | Reads the file content and manages communication with the server. |
| EXIT |  | Programs exist automatically after a successful or failed connection. |

### Server

|  |  |  |
| --- | --- | --- |
| **Field** | **Value** | **Description** |
| Shift Value | integer | Reads the shift value and shifts the context received by this value. |
| Handle Signal | interger | Handle SIGINT for graceful shutdown. |

# **Functions**

## Client

|  |  |
| --- | --- |
| **Function** | **Description** |
| main | Validates arguments, reads the file, and communicates with the server. |
| validate\_arguments | Checks command-line arguments for validity. |
| prepare\_filename | Copies and sanitizes the provided file name. |
| open\_file | Opens the file safely for reading. |
| get\_file\_size | Determines the size of the file. |
| read\_file\_content | Reads file content into memory. |
| close\_file | Closes the file safely. |
| create\_client\_socket | Creates a UNIX domain socket for the client. |
| connect\_to\_server | Establishes a connection to the server. |
| send\_message\_to\_server | Sends file content to the server. |
| receive\_server\_response | Receives and displays the server’s encrypted response. |
| close\_socket | Closes the client socket. |

## Server

|  |  |
| --- | --- |
| **Function** | **Description** |
| main | Validates arguments, initializes the server, and handles connections. |
| validate\_arguments | Parses and validates command-line arguments for shift value. |
| setup\_server\_socket | Sets up the server’s UNIX domain socket and starts listening. |
| accept\_client\_connections | Accepts client connections and processes messages. |
| process\_client\_message | Receives, encrypts, and sends back client messages. |
| caesar\_encrypt | Encrypts a given message using the Caesar cipher. |
| handle\_signal | Handles CTRL+C signal to perform a graceful shutdown. |
| cleanup | Closes the socket and removes the socket file. |

# **States**

## Client

|  |  |
| --- | --- |
| **State** | **Description** |
| START | Start client program and read the input file. |
| VALIDATE | Check file name and ensure all arguments are correct. |
| OPEN FILE | Get the file ready to write its content to the buffer. |
| READ | Read file contents to the Buffer. |
| INIT\_SOCKET | Set up the UNIX domain socket. |
| CONNECT | Establish a connection to the server. |
| SEND | Transmit file content to the server. |
| RECEIVE | Receive and display the encrypted response. |
| CLEANUP/ CLOSE SOCKET | Close the connection and free memory. |

## Server

|  |  |
| --- | --- |
| **State** | **Description** |
| START | Start server program and read the shift value. |
| VALIDATE | Check shift value and ensure all arguments are correct. |
| INIT\_SOCKET | Set up the UNIX domain socket. |
| LISTEN | Wait for client connections. |
| ACCEPT | Accept incoming client connection. |
| PROCESS | Encrypt incoming messages and send back the response. |
| CLEANUP | Release resources and exit. |

# **State Table**

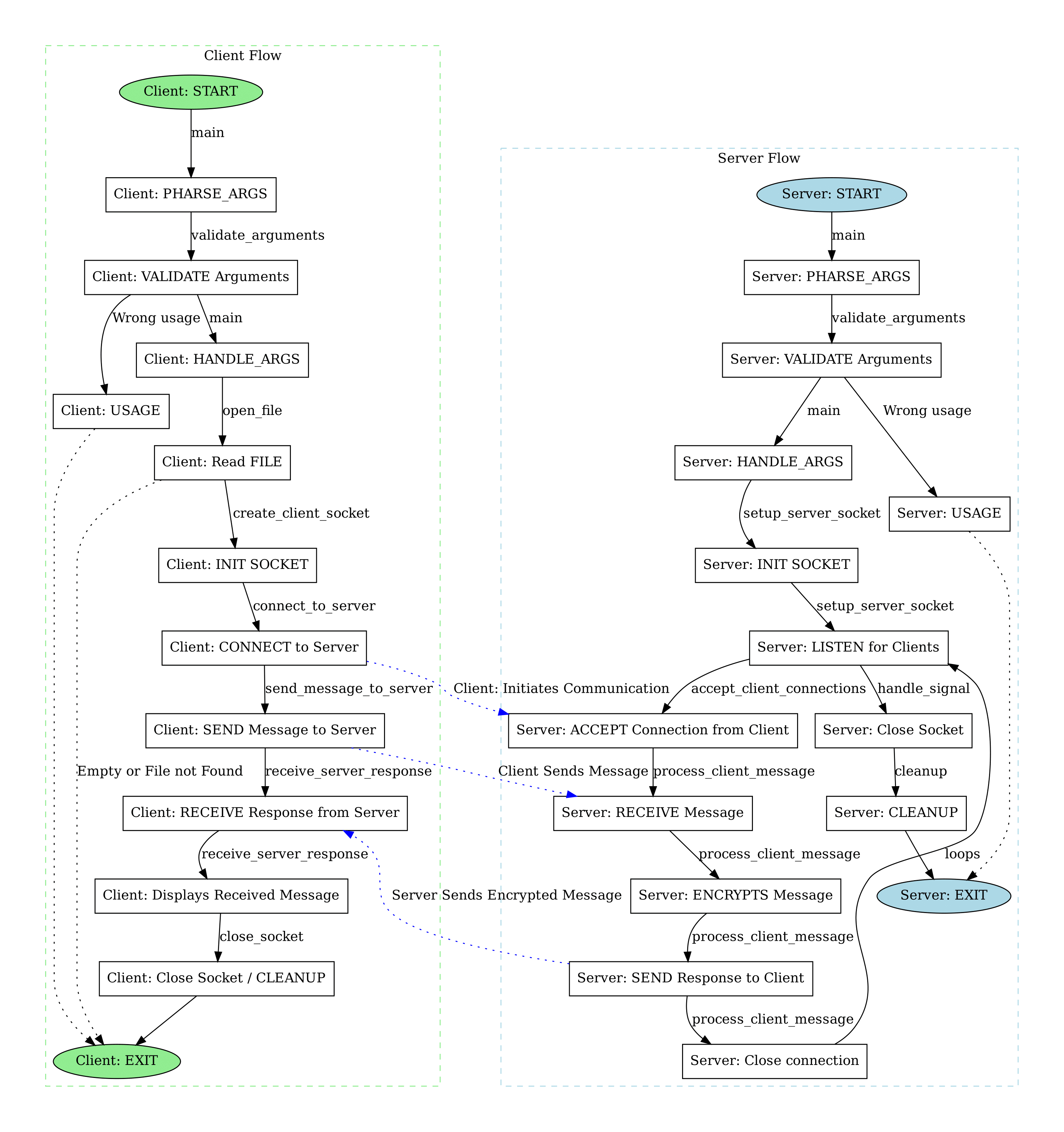
## Client

|  |  |  |
| --- | --- | --- |
| **From State** | **To State** | **Function** |
| START | VALIDATE | validate\_arguments |
| VALIDATE | OPEN FILE | prepare\_filename |
| OPEN FILE | READ | open\_file |
| READ | INIT\_SOCKET | create\_client\_socket |
| INIT\_SOCKET | CONNECT | connect\_to\_server |
| CONNECT | SEND | send\_message\_to\_server |
| SEND | RECEIVE | receive\_server\_response |
| RECEIVE | CLEANUP/ CLOSE SOCKET | close\_socket |

## Server

|  |  |  |
| --- | --- | --- |
| **From State** | **To State** | **Description** |
| START | VALIDATE | validate\_arguments |
| VALIDATE | INIT\_SOCKET | setup\_server\_socket |
| INIT\_SOCKET | LISTEN | setup\_server\_socket |
| LISTEN | ACCEPT | accept\_client\_connections |
| ACCEPT | PROCESS | process\_client\_message |
| PROCESS | LISTEN | process\_client\_message |
| PROCESS | CLEANUP | handle\_signal |
| CLEANUP | EXIT | cleanup |

# **State Transition Diagram**



# **Pseudocode**

## Client

## validate\_arguments:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| argument\_count | integer | The passed Arguments |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

validate\_arguments:

if the number of arguments is not 2:

print error message

terminate with failure status

## prepare\_filename:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| dest | character string | A string to store the resulting filename |
| src | character string | A string containing the original filename |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

prepare\_filename:

copy up to (FILE\_NAME\_SIZE - 1) characters from source to destination

set the last character of destination to null-terminator

## **open\_file:**

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| filename | character string | A string containing the name of the file to be opened |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| filename | A reference or handle to the opened file (used for interacting with the file) |

### Pseudo Code

open\_file:

try to open the file in read mode using the provided filename

if the file cannot be opened:

print an error message to output

terminate with failure status

return the file handle or reference

## **get\_file\_size:**

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| file | A reference or handle | Determine file size |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| integer or long | Representing the size of the file in bytes. |

### Pseudo Code

get\_file\_size:

find file size

return the file size

## read\_file\_content:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| file | A reference or handle | To read the file. |
| file\_size | unsigned integer or long | To know now many bytes to read. |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| string | File's content, null-terminated |

### Pseudo Code

read\_file\_content:

create a container (e.g., string or buffer) to hold the file content, sized to fit the file's size

if the container cannot be created:

print an error message

terminate with failure status

read the entire file into the container

ensure the container has an end-of-content marker (if required by the system)

return the container containing the file's content

## close\_file:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| file | A reference or handle | Need to be closed. |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

close\_file:

if the file is valid:

close the file

## create\_client\_socket:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| None |  |  |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| whole number | Represents the created socket, used for communication |

### Pseudo Code

create\_client\_socket:

attempt to create a socket with appropriate settings (e.g., communication type and protocol)

if socket creation fails:

print an error message

terminate with failure status

return the socket descriptor or handle

## connect\_to\_server:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| client\_socket | A reference or handle | Handle to the client-side communication channel |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| NA |  |

### Pseudo Code

connect\_to\_server:

create an address structure to specify server details

initialize the address structure with default values

set the address family to match the communication protocol

set the server path or identifier

attempt to connect to the server using the provided socket and address

if the connection fails:

print an error message

close the client socket to release resources

terminate with failure status

print a message indicating the connection was successful

## send\_message\_to\_server:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| client\_socket | A reference or handle | **Handle to the client-side communication channel** |
| message | string | **Message to be sent** |
| size | whole number | **Size of the message** |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

send\_message\_to\_server:

attempt to send the specified message through the client socket

if the message cannot be sent:

print an error message

close the client socket to release resources

terminate with failure status

print a message indicating the message was successfully sent

## receive\_server\_response:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| client\_socket | A reference or handle | handle representing the client socket used to receive data |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

receive\_server\_response:

create a buffer to hold incoming data

initialize a flag to indicate if receiving is complete

print a message indicating the start of response reception

while not done receiving:

receive data from the server into the buffer

if data is received:

add an end-of-content marker to the buffer

display the received data

if the received data is less than the buffer size:

mark receiving as complete

if no data is received:

mark receiving as complete and print

if an error occurs:

print an error message

mark receiving as complete

print a message indicating disconnection from the server

## close\_socket:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| client\_socket | A reference or handle | Identifier for the communication channel to be terminated |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

close\_socket:

close the communication channel represented by the socket

## Server

## validate\_arguments:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| client\_socket | A reference or handle | Identifier for the communication channel to be terminated |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

validate\_arguments:

if the number of arguments is not 2:

print an error message

terminate with failure status

attempt to convert the second argument to a number (shift value)

if the conversion is invalid (non-integer input):

print an error message "Error: Shift must be an integer"

terminate with failure status

## setup\_server\_socket:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| None |  |  |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| whole number | Represents the server socket, used for accepting connections. |

### Pseudo Code

setup\_server\_socket:

create a socket for server communication with appropriate settings (e.g., communication type and protocol)

if socket creation fails:

print an error message

terminate with failure status

create an address structure to store server details

initialize the address structure with default values

set the address family to match the communication protocol

set the server path or identifier

if the server path already exists:

remove the existing server path

attempt to bind the server socket to the address

if binding fails:

print an error message

clean up resources and terminate with failure status

attempt to listen for incoming client connections

if listening fails:

print an error message

clean up resources and terminate with failure status

print a message indicating the server is listening

return the server socket descriptor or handle

## 

## accept\_client\_connections:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| server\_socket | A reference or handle | Handle for the server's communication channel that listens for incoming connections. |
| shift | integer | A value to be applied to the client's message during processing. |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

accept\_client\_connections:

enter an infinite loop to continuously wait for client connections

print a message indicating the server is waiting for a client

attempt to accept a client connection

if the client connection fails:

print an error message

continue to the next iteration to wait for another client

print a message indicating the client is connected

process the client's message using the provided shift value

close the client socket to terminate the connection

print a message indicating the client has disconnected

## process\_client\_message:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| server\_socket | A reference or handle | Handle for the server's communication channel that listens for incoming connections. |
| shift | integer | A value to be applied to the client's message during processing. |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

process\_client\_message:

create a buffer to hold incoming data

initialize a flag to indicate when receiving is complete

enter a loop to continuously receive data from the client

while not done receiving:

clear the buffer

receive data from the client into the buffer

if data is received:

add an end-of-content marker to the buffer

apply a transformation (e.g., encryption) to the data using the provided shift value

send the transformed data back to the client

if the received data is less than the buffer size:

mark receiving as complete

if no data is received:

mark receiving as complete

if an error occurs during receiving:

print an error message

mark receiving as complete

print a message indicating that the process is complete and data has been sent

## caesar\_encrypt:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| message | character string | Characters representing the message to be encrypted. |
| shift | integer | A value to be applied to the client's message during processing. |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

function caesar\_encrypt:

loop through each character of the message until the end of the string

if the character is a lowercase letter:

apply the shift to the character within the range of lowercase letters

if the character is an uppercase letter:

apply the shift to the character within the range of uppercase letters

return the encrypted message

## handle\_signal:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| signal | integer | Interrupt the server. |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

handle\_signal:

if the received signal is SIGINT (Ctrl+C):

print a message indicating the signal was caught and the program is shutting down

clean up resources

terminate the program successfully

## cleanup:

### Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| None |  |  |

### Return

|  |  |
| --- | --- |
| **Value** | **Reason** |
| None |  |

### Pseudo Code

cleanup:

if the server socket is valid:

close the server socket

print a message indicating the socket was closed

attempt to remove the socket file:

if the removal is successful:

print a message indicating the socket file was unlinked

else:

print an error message indicating failure to remove the socket file