COMP 7005
Assignment 2

Design

Winston Nguyen
A01297231
Feb 6th, 2024

Purpose

States

Client

State	Description
PARSE_ARGS	Parse command line arguments
HANDLE_ARGS	Verify and convert the command line arguments for use
HANDLE_ERROR	Display an error message when the command line arguments have an issue
SOCKET_CREATED	Create the socket ready to be binded.
SOCKET_BINDED	Binds the socket ready to send/receive connections.
SEND_DATA	Send data over the connection.
LISTEN	Listen for data to be returned over the connection.
CONNECTION_RECEIVED	Receive a connection and be ready to handle it.
HANDLE_CONNECTION	Handle a connection and parse through ready for use.
CLEANUP	Cleanup before exit
DISPLAY_MESSAGE	Display any messages to the screen

Server

State	Description
HANDLE_ERROR	Display an error message when the command line arguments have an issue
SOCKET_CREATED	Create the socket ready to be binded.
SOCKET_BINDED	Binds the socket ready to listen/receive connections.

LISTENING	Listen for connections to be received.
CONNECTION_RECEIVED	Receive a connection and be ready to handle it.
HANDLE_CONNECTION	Handle a connection and parse through ready for use.
CLEANUP	Cleanup before exit

State Table

Client

From State	To State	Function
START	PARSE_ARGS	parse_arguments
PARSE_ARGS	HANDLE_ARGS	handle_arguments
PARSE_ARGS	HANDLE_ERROR	handle_error
HANDLE_ARGS	SOCKET_CREATED	create_socket
HANDLE_ARGS	HANDLE_ERROR	handle_error
SOCKET_CREATED	SOCKET_BINDED	bind_socket
SOCKET_CREATED	HANDLE_ERROR	handle_error
SOCKET_BINDED	SEND_DATA	send_data
SOCKET_BINDED	HANDLE_ERROR	handle_error
SEND_DATA	LISTEN	listen_connection
SEND_DATA	HANDLE_ERROR	handle_error
LISTEN	CONNECTION_RECEIVED	receive_connection
CONNECTION_RECEIVED	HANDLE_CONNECTION	handle_connection
HANDLE_CONNECTION	DISPLAY_MESSAGE	display_message
HANDLE_CONNECTION	HANDLE_ERROR	handle_error

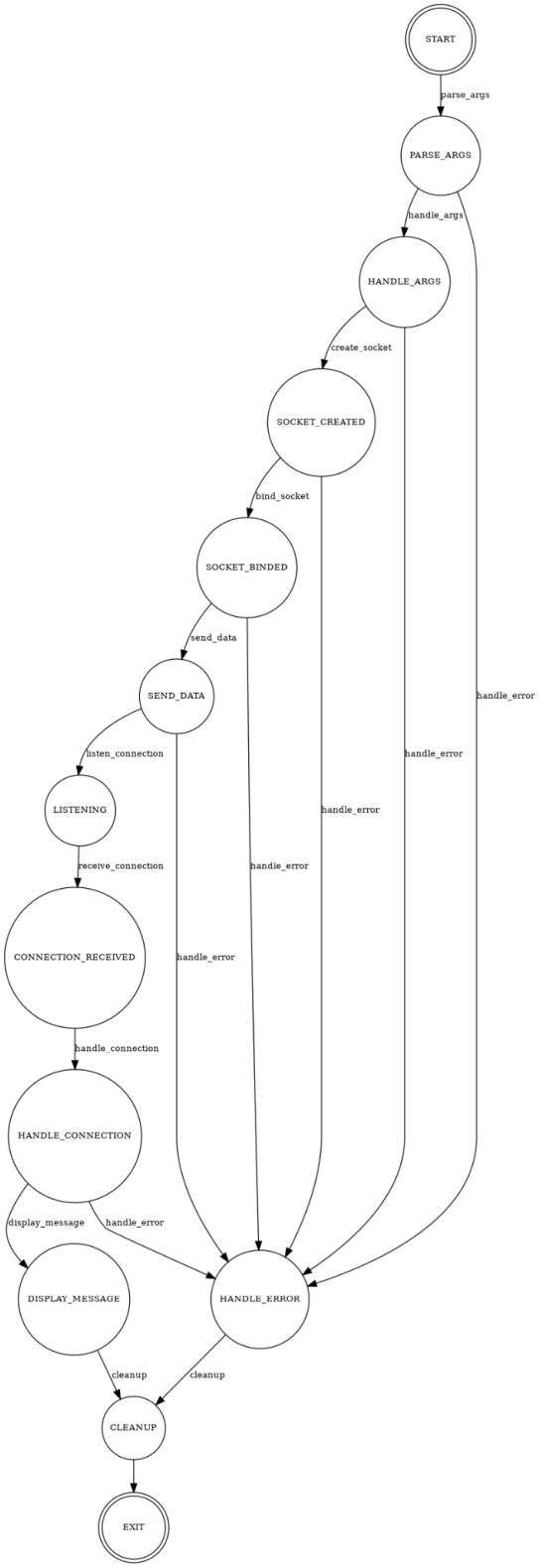
DISPLAY_MESSAGE	CLEANUP	cleanup
HANDLE_ERROR	CLEANUP	cleanup
CLEANUP	EXIT	

Server

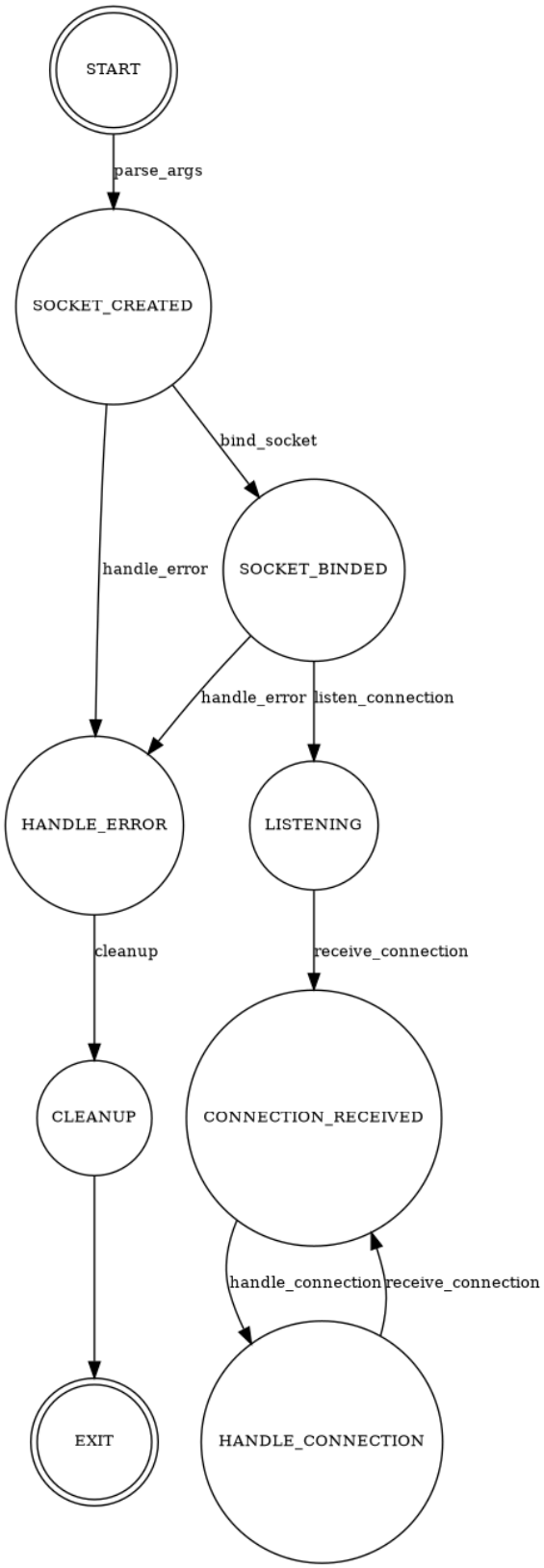
From State	To State	Function
START	SOCKET_CREATED	create_socket
SOCKET_CREATED	SOCKET_BINDED	bind_socket
SOCKET_CREATED	HANDLE_ERROR	handle_error
SOCKET_BINDED	LISTENING	listen_connection
SOCKET_BINDED	HANDLE_ERROR	handle_error
LISTENING	CONNECTION_RECEIVED	receive_connection
CONNECTION_RECEIVED	HANDLE_CONNECTION	handle_connection
HANDLE_CONNECTION	CONNECTION_RECEIVE	receive_connection
HANDLE_ERROR	CLEANUP	cleanup
CLEANUP	EXIT	

State Transition Diagram

Client



Server


START
parse_args
SOCKET_CREATED
bind_socket
handle_error
SOCKET_BINDED
handle_error
listen_connection
HANDLE_ERROR
LISTENING
cleanup
receive_connection
CLEANUP
CONNECTION_RECEIVED
handle_connection
receive_connection
EXIT
HANDLE_CONNECTION

Data Types

Arguments

Purpose: To pass data across functions

Field	Type	Description
args	string[]	The arguments
file_name	string	The name of the file to open and read
ip_address	string	The ip address of the server to connect with

Context

Field	Type	Description
arguments	Arguments	The command line arguments
ip	string	The ip of the server
file	string	The name of the file with data
port	string	The port the server connects with
connection	socket	The connection to send/receive data over

Functions

Function	Description
parse_arguments	Parse the command line arguments
handle_arguments	Verify the command line arguments
create_socket	Create a socket server to connect to/with
connect_client	Connect the client to the socket
send_message	Send an encoded message across the connection
receieve_response	Response a response from the server/client.
read_file	Read a files contents to be sent to the server

replace_new_lines	Replace the new line characters in the file to as spaces.
is_ipv4	Check if the ip address set is an IPv4 address or an IPv6 address.
handle_error	Print any errors that occur.
display_message	Display any messages to the user.
cleanup	Cleanup any connections and close them properly.
listen_connections	Listen for any incoming connections to receive data from.
accepet_connection	Accept a connection once found.
handle_data	Handle the data received from the connection by calling other functions
send_response	Send a response to the connection once data has been handled.
get_words	Get the words in a string and place them into array for manipulation.
remove_whitespace	Remove extra whitespaces between words in a string.
get_word_count	Get the number of words in an array.
get_char_count	Get the number of characters in the array.
get_char_freq	Get the number of times each character appears in the array.
sort_dict	Sort a dictionary of character frequencies alphabetically by their keys.
format_response	Format the data analyzed into a user-friendly string to send to the client.

Pseudocode

check_args

Parameters

Parameter	Description
args	The command line arguments passed to main

Return

- Nothing

Pseudo Code

```
If length of args is not a valid length (2):
     call handle_error
If passed in argument is not a valid file ext (.txt):
     call handle_error
```

handle_args

Parameters

Parameter	Description
args	The command line arguments passed to main

Return

- Nothing

Pseudo Code

```
Prepare filename for other functions to use
If error:
     call handle_error
```

create_socket

Parameters

Parameter	Description

Return

- Nothing

Pseudo Code

```
Create a socket ready for IPv4 and IPv6 addresses
If failure to create socket:
     call handle_error
```

connect_client

Parameters

Parameter	Description
server_host	The address of the server to connect with.
server_port	The port the server uses to connect with.

Return

- Nothing

Code

```
Connect client to socket using server_host and port
If failure to connect to path:
     call handle_error
```

send_message

Parameters

Parameter	Description
words	A string of data to send to the server

Return

- Nothing

Pseudo Code

```
Encode words to send
Send encoded words via the socket
If failure to send words:
     Throw error
```

receieve_response

Parameters

Parameter	Description

Return

- Nothing

Pseudo Code

```
Wait for response from server
Print response
If failure to receive response:
     Throw error
```

close_socket_client

Parameters

Parameter	Description
client	The client connection to be closed.

Return

- Nothing

Code

```
Close the client
If failure to close the client
     Throw error
```

read_file

Parameters

Parameter	Description
file_name	The name of the file to read.

Return

- String: words read inside the given file_name

Pseudo Code

```
Open the file using file_name as a read
     Read the content
     Call replace_new_lines
```

replace_new_lines

Parameters

Parameter	Description
text_data	The string contains words from a text file.

Return

- String: text_data with the new lines replaced.

Pseudo Code

```
Replace all newlines inside of text_data with spaces
If failure to replace lines:
     Throw error
```

is_ipv4

Parameters

Parameter	Description
ip_str	The string of the IP address the client will use to connect to the server.

Return

- Boolean:

Code

```
If ip_str is IPv4:
     return True
If ip_str is IPv6:
     return False
Else:
     call handle_error
```

handle_error

Parameters

Parameter	Description
err_message	The message of the error to print.

Return

- Nothing

Code

```
print err_message
call cleanup
```

display_message

Parameters

Parameter	Description
message	The message to print to the screen

Return

- Nothing

Code

```
print message
call cleanup
```

cleanup

Parameters

Parameter	Description
success	A boolean of the program's exit type.

Return

- Nothing

Code

```
Close the connection
If success:
     Exit 0
Else:
     Exit 1
```

listen_connections

Parameters

Parameter	Description

Return

- Nothing

Pseudo Code

```
Listen for incoming connections
Wait until a connection is found
If failure to connect to a connection:
     Throw error
```

accept_connection

Parameters

Parameter	Description

Return

- Bytes: encoded data from connection

Pseudo Code

```
Readable = [] // readable connections
Writable = [] // writable connections
Accept found connections
Add connection to readables when ready
If connection is readable:
     Receive the connections data
     Add connection to Writable

If data is ready to write:
     call handle_data
     Write data to writable connection

If failure to receive data:
     call handle_error
```

handle_data

Parameters

Parameter	Description
data	The data sent over connections in bytes.

Return

- String: data analyzed and formatted for return over connection

Pseudo Code

```
Call get_words
Call get_word_count
Call get_char_count
Call get_char_freq
Call sort_dict
Call format_response
Return formatted_response
```

send_response

Parameters

Parameter	Description
response	The formatted response to send to the client/accepted connection.

Return

- Nothing

Pseudo Code

```
Encode the response
Send the encoded response through the connection.
If failure to send response:
     Throw error
```

get_words

Parameters

Parameter	Description
word_string	The string of words received from the client/connection.

Return

- Array: list of words

Pseudo Code

```
Call remove_whitespace
Separate words inside of word_string by spaces
Remove empty strings in the list
Return list
```

remove_whitespace

Parameters

Parameter	Description

word_string	A string of words.

Return

- String: with extra whitespaces removed

Pseudo Code

```
Replace consecutive whitespaces with a single space.
If failure to remove whitespaces:
     Throw error
```

get_word_count

Parameters

Parameter	Description
words	An array of strings.

Return

- Int: the number of words in the list

Pseudo Code

```
Get the length of words
Return
```

get_char_count

Parameters

Parameter	Description
words	An array of strings.

Return

- Int: the number of characters total.

Pseudo Code

```
Set count to 0
For each string in words:
     Increase count by the length of the string
Return count
```

get_char_freq

Parameters

Parameter	Description
words	An array of strings.

Return

- Dictionary: holding the number of times a character is found

Pseudo Code

```
Create a dictionary

// List comprehension in python
Set each string to lowercase in words

Join the words to a single string
For each char in words:
     If the char does not exists in the dictionary:
          Set dictionary[char] = 0
     Else:
          Increase dictionary[char] by one
Return dictionary
```

sort_dict

Parameters

Parameter	Description
char_freq	A dictionary of character frequencies.

Return

- Dictionary: sorted by keys

Pseudo Code

```
Get the keys inside the dictionary
Sort dictionary in ascending order using keys
Return sorted dictionary
```

format_response

Parameters

Parameter	Description
word_count	The number of words in the connection’s data string.
char_count	The number of characters in the connection’s data string.
char_freq	Each character's frequencies found in the connection’s data string.

Return

- String: a formatted response.

Pseudo Code

```
Format word_count, char_count, char_freq into a single string
Return formatted string
```