Client SMTP in C

Instructions and documentations

Liviu Arsenescu

16.06.2024

Contents

1 Introduction **2**

2 Client usage **2**
2.1 Synopsis . . . 2
2.2 Description . . . 2
2.3 Example . . . 2
2.4 Compile the program . . . 2

3 State Machine and Implementation **3**
3.1 State Machine . . . 3
3.2 Implementation . . . 3
3.2.1 Includes and Definitions . . . 3
3.2.2 Function Declarations . . . 4
3.2.3 SMTP State Machine . . . 4
3.2.4 Main Function . . . 4
3.2.5 Command Line Arguments . . . 4
3.2.6 Argument Parsing . . . 5
3.2.7 File Handling . . . 5
3.2.8 SMTP Communication Loop . . . 5
3.2.9 tcp_connect Function . . . 5

4 Tests **5**

1 Introduction

Welcome to the documentation for the Simple SMTP Client, a project developed for the Networking course at HE-Arc Ingénierie. This project showcases the practical application of networking concepts by implementing a basic SMTP client, designed to facilitate the understanding of email protocols and client-server communication.

2 Client usage

2.1 Synopsis

```
bin/client_smtp
    <sender email>
    <subject>
    <message file>
    <mail server>
    <reciever email>
    [<port>]
```

2.2 Description

`<sender email>` - Email address of the sender.
`<subject>` - Subject of the email.
`<message file>` - Path to the file containing the message.
`<mail server>` - Address of the mail server.
`<reciever email>` - Email address of the reciever.
`[<port>]` - (optional) Port number of the mail server. Default is 25.

2.3 Example

```
bin/client_smtp
    liviu-andrei.arsenescu@he-arc.ch
    "Some Subject"
    mail_body.txt
    smtp.alphanet.ch
    liviu-andrei.arsenescu@he-arc.ch
    587
```

2.4 Compile the program

To compile the program, you can use the Makefile provided:

```
make
```

The program executable is situated in `./bin`

3 State Machine and Implementation

3.1 State Machine

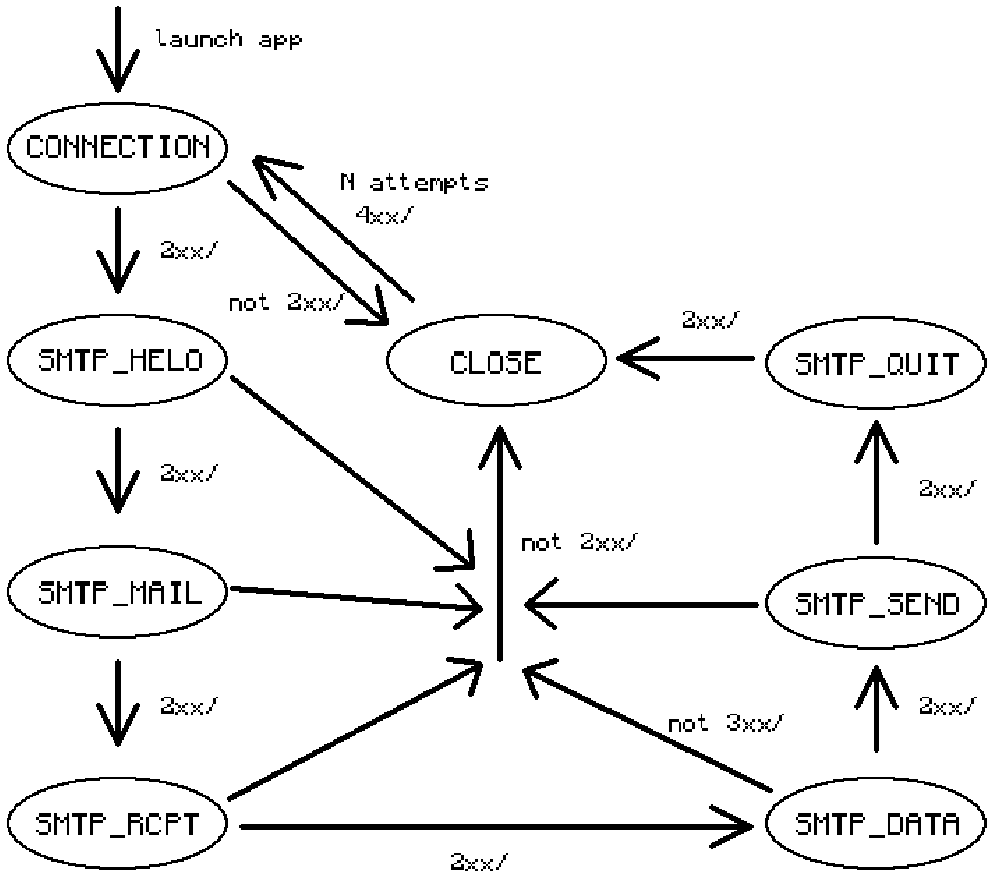

Figure 1: State Machine

3.2 Implementation

3.2.1 Includes and Definitions

```
1     #include <stdio.h>
2     #include <unistd.h>
3     #include <stdlib.h>
4     #include <string.h>
5     #include <netdb.h>
6     #include <sys/types.h>
7     #include <sys/socket.h>
8     #include <sysexits.h>
```

`#include` Directives: These include standard C libraries (`stdio.h`, `unistd.h`, `stdlib.h`, `string.h`, `netdb.h`, `sys/types.h`, `sys/socket.h`) and `sysexits.h` for exit status codes.

```
#define DEFAULT_PORT "25"
#define MAX_ATTEMPTS 5
#define WAIT_TIME 5
```

Constants

- **DEFAULT_PORT**: Default port number for SMTP (port 25)
- **MAX_ATTEMPTS**: Maximum number of connection attempts
- **WAIT_TIME**: Time to wait between connection attempts (in seconds)

3.2.2 Function Declarations

```
static FILE *tcp_connect(const char *hostname, const char
    *port);
```

tcp_connect establishes a TCP connection to a specified hostname and port, returning a FILE* for socket communication.

3.2.3 SMTP State Machine

```
typedef enum {
    CONNECTION,
    SMTP_HELO,
    SMTP_MAIL,
    SMTP_RCPT,
    SMTP_DATA,
    SMTP_SEND,
    SMTP_QUIT,
    CLOSE,
} smtp_state_t;
```

Defines different states **(smtp_state_t)** for the SMTP client to manage the sequence of SMTP commands required to send an email

3.2.4 Main Function

```
int main(int argc, char **argv)
```

Entry point of the program where command-line arguments are validated and SMTP communication is managed

3.2.5 Command Line Arguments

```
if(argc < 6 || argc > 7) {
    fprintf(stderr, "[!] Usage: %s <sender email> <subject>
        <message file> <mail server> <receiver email>
        [<port>]\n", argv[0]);
    exit(EX_USAGE);
}
```

Argument Validation: Ensures correct usage with necessary command-line arguments (sender email, subject, message file, mail server, receiver email, optional port)

3.2.6 Argument Parsing

```
    char *sender = argv[1];
    char *subject = argv[2];
    char *message_file = argv[3];
    char *mail_server = argv[4];
    char *receiver = argv[5];
    char *port = (argc == 7) ? argv[6] : DEFAULT_PORT;
```

Argument Assignment: Copies command-line arguments into variables for easier access and validation

3.2.7 File Handling

```
    FILE* mail_body = fopen(message_file, "r");
    if(mail_body == NULL) {
        fprintf(stderr, "[!] Could not open message file: %s\n",
            message_file);
        exit(EX_NOINPUT);
    }
```

File Opening: Opens the message file for reading. If the file cannot be opened, it exits with **EX_NOINPUT**

3.2.8 SMTP Communication Loop

```
    smtp_state_t state = CONNECTION;
    // Loop to manage SMTP states and communication
    while(1) {
        switch(state) {
            // Each case handles a specific SMTP command/response
                sequence
        }
    }
```

SMTP Communication Loop: Continuously loops through SMTP states **(CONNECTION, SMTP_HELO, SMTP_MAIL, etc.)** to establish connection, send email data, and handle server responses

3.2.9 tcp_connect Function

```
    static FILE *tcp_connect(const char *hostname, const char
        *port)
```

4 Tests