BRNO, UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Network Applications and Network Administration – ISA project

Simple SMTP client

November 9, 2015

Martin Kacmarcik (xkacma03)

Contents

1 Introduction to this document

This report is a documentation for my ISA project - simple SMTP client. Document will cover the description of how I implemented this project using RFC 2821. As a simple characterization of the application, if input file is specified in the right format, it will send emails to receivers to the SMTP server. All the information are based on study of the RFC 2821 that describe SMTP protocol[1].

2 Introduction to the SMTP protocol

As mentioned, SMTP protocol is defined in RFC 2821. It was also defined in RFC 821 but in this project we are implementing the advanced SMTP protocol. SMTP stands for Simple Mail Transfer Protocol and is a way to implement Internet electronic mail transport. SMTP client first start the connection using sockets. It connects to the SMTP server using server's IP address (or hostname, if DNS can resolve it) on port that can vary (although 25 is default one). Then, simply said, using text commands defined in RFC, client can send an email to the destination mailbox. Application works with both IPv4 and IPv6 addresses. The difference between IPv4 and IPv6 socket implementation is just using a little bit different structures and functions to set these structures. Other usage of the sockets is the same for both IP variations.

3 Application draft

3.1 Introduction this chapter

In this section I will describe how I designed the application, what language I was using etc.

3.2 Language choice

For the language - I used the classic C. The reason why I didn't use the advanced C++ was simple - I am just so familiar and fast in C language and I didn't want to waste time learning new syntax. Also I had previous projects using sockets in C so I could just use my code for this projects as well. For the debugging the problems I also often used Valgrind and log in /var/log/mail.log which helped me a lot. The client was implemented and tested on reference Linux virtual machine with Ubuntu Linux. For translation I am using gnu99 compiler - just because c99 didn't worked properly with my functions and gnu99 was smooth.

3.3 Short design description

The design of the application it's self has several steps. First I needed to parse the file it's self. I am using my own function that will first count number of characters, allocate proper space and assign content of the file into my memory. After I have the content of the file in my memory I am calling my parsing functions, that will split string into lines and allocate array of lines. With this knowledge I can call function that handle sockets and the transactions itself. In the function, I try to connect to the given server and in while cycle parse the line into message and array full of recipients email addresses. With this knowledge I can start the transaction.

3.4 Transaction description

The transaction is defined in RFC 2821. It start when client connects to the server. Server response with Welcome message. After, the client is supposed to say "EHLO domain" (or HELLO in old SMTP version) and acknowledge the server that the transaction begin. If everything is OK, server

will response with message ”250”. Then the client will define from whom is the mail sent, using command ”MAIL FROM:<email_address>”. Response from the server should be again 250 OK. Next, the server expect recipients. They are defined using command ”RCPT TO:<email_address>”. If recipient doesn't exist, server will return error number 550. If does, server will return again 250 OK. This way client can send multiple recipients that shall receive the same message. After the definition of the user is finished, client is suppose to acknowledge server that DATA transaction beings. This is done by sending ”DATA” command. If there was atleast one valid recipient, server will response with ”354 Start mail input; end with <CRLF>. <CRLF >” message. Now server is expecting the actual message. In my case I am sending only the string itself and nothing more. Then the DATA section is ended by client sending ”<CRLF >. <CRLF >” message. Again, if all worked out, server will response 250 OK. In our simple SMTP client, after the message is in server's queue we end the transaction using QUIT command. Then it is needed to wait for the server to response with 221 message, that acknowledge us that we can close the socket and everything is set. As said, next step is to close the connection by closing socket using socket descriptor. I am also trying to send as much messages as possible and only print on stderr errors (e.g. when recipient doesn't exist).

3.5 Problem with signals and ending everything properly

There was a problem when terminating signal came and in the instruction is defined that we need to end connection properly after signal is sent. I implemented this problem using simple global variable that is set to true when we are in data section and to false any other time. When signal is received, variable is checked and if it's true, the actual message is send first with ending dot message after that. Then I quit the transaction using QUIT command.

3.6 Global variables

Because of the signal handling, to clear memory properly, I am using global variables to have access to pointers that needs to be free as well as to the socket descriptor and file pointer.

4 Usage and examples

4.1 Usage

Application doesn't need any special rights. Usage is: usage: ./smtpklient [-h] [-a IP] [-p port] -i file [-w seconds]
Where:

- h will print help
- a param is volatile address param, 127.0.0.1 is default
- p is port number, volatile, 25 is default
- i is mandratory and leads to config file with rcpt and message
- w is volatile param that will keep connection alive x seconds where x is subparam of w

4.2 Examples of use

- IPv4 address:
  ./smtpklient -a 127.0.0.1 -p 28 -i inputFile.txt -w 10
  –>Result: In mailbox there are sent mail if everything went correctly

- IPv6 address:
  ./smtpklient -a ::1 -p 28 -i inputFile.txt -w 10
  –>Result: In mailbox there are sent mail if everything went correctly
- NonExistingFile
  ./smtpklient -i nonExistingFile
  –>Error while opening file...
- Print help:
  ./smtpklient -h
  –>Help for simple SMTP client by Martin Kacmarcik - xkacma03 For FIT VUTBR - ISA course 2015(...)

5 Conclusion

Program sends strings and receive defined by RFC 2821 using BSD sockets to and from the destination mail server. This way, emails to various recipients can be send. Application is implemented in C language. Application was successfully tested on reference Ubuntu virtual machine.

6 Literature

[1] J. Klensin, Editor, "Simple Mail Transfer Protocol", IETF, RFC 2821, April 2001, [Online]. Available: `https://www.ietf.org/rfc/rfc2821.txt`