POP3 Client with TLS

2021/2020

Network Applications and Network Administration

Project documentation

Roman Janiczek (xjanic25)

November 15, 2021

Contents

1 Introduction **2**
- 1.1 Protocol POP3 . . . 3
  - 1.1.1 Authorization state . . . 3
  - 1.1.2 Transaction state . . . 3
  - 1.1.3 Update state . . . 4
  - 1.1.4 Adding support for TLS . . . 4

2 Implementation **5**
- 2.1 Main . . . 5
- 2.2 ArgumentParser class . . . 5
- 2.3 POP3 class . . . 5
  - 2.3.1 startConnection() . . . 5
  - 2.3.2 login() . . . 6
  - 2.3.3 retrieveAllMessages() . . . 6
  - 2.3.4 quit() . . . 6
- 2.4 Connection class . . . 6
- 2.5 FileManipulator class . . . 6

3 How to use **7**
- 3.1 Usage . . . 7
  - 3.1.1 Required arguments . . . 7
  - 3.1.2 Optional arguments . . . 7
- 3.2 Source files for the project . . . 8

4 Sources and literature **9**

1 Introduction

This project and documentation focuses on the implementation of a simple mail client (for receive only) using Post Office Protocol v3 (POP3) as described by RFC1939 and its extension for TLS (POP3s, STARTTLS) specified by RFC2595.

1.1 Protocol POP3

For communication with the server, POP3 uses reliable Transmission Control Protocol (TCP). The default port for POP3 is 110.

POP3 consist of 3 states:

- Authorization state
- Transaction state
- Update state

The communication process is as follows. First, we establish the connection with the server, which puts us in an Authorization state and awaits login credentials. Second, after successful login, we get to the Transaction state, where it's possible to retrieve or delete messages from the server. And lastly, at the client's request, we move to the Update state where the changes are saved - delete marked messages.

POP3 defines commands that the client can use at each state; every command has to be ended with `\r\n`.

Responses from the are either:

- one-line response ended with `\r\n`
- multi-line response ended with `\r\n.\r\n`

1.1.1 Authorization state

After a successful connection to the server - confirmed by message in `+OK <msg>` format from the server, we are in Authorization state. The authorization state accepts the following commands `USER`, `PASS`, `APOP`, and `QUIT`.

1.1.2 Transaction state

If the user successfully authorizes to the POP3 server, the connection moves to the transaction state.

Here, the following commands are available:

- `STAT` - gives information about the number of mails available and their size
- `LIST` <id>
- `RETR` <id> - returns message with given number to the client
- `DELE` <id> - marks message with given number for deletion
- `NOOP`
- `RSET` - removes deleted flag from any messages marked as deleted during the current session

Thanks to these, this state allows manipulation with the mailbox.

1.1.3 Update state

This state is initiated by the client using the command `QUIT`.

`QUIT` command logs out user and saves all changes made - marked as read, deleted messages.

1.1.4 Adding support for TLS

POP3 is an unsecured protocol in its basic form - a third party can catch and read all client-server communication. The TLS extension for POP3 amends this.

There are two options for this:

- New port created only for secure communication using TLS (default is 995)
- Additional command `STLS` in Authorization state tells the server to initiate a secure connection on port 110.

2 Implementation

Language: `C++`
Additional libraries:

- `openssl`
- `getopt` (included in `util-linux`)

2.1 Main

The program starts by creating an instance of `ArgumentParser` class in `main.cpp`, which then uses the `parse` method to process arguments specified by the user.

If arguments are parsed without problem, the program creates an instance of the `POP3` class, which will act as a POP3 client and passes it instance of `ArgumentParser` class using method `passArgs`. After this, method `run` of the `POP3` class is called to start the client.

2.2 ArgumentParser class

This class handles all of the argument processing - parsing and validation.

The main method of this class is method `parse`, which makes use of `getopt` functionality to parse arguments.

Validation is handled in method `validate`, which is called at the end of the `parse` method. The `validate` method does the basic checks to confirm that arguments make sense and loads credentials from the authorization file.

In case of any problems encountered in this class, error details are printed out on `stderr`. The user is also informed on `stdout` in case of problems with CAfile, CAdir, and an authfile.

2.3 POP3 class

The main logic of the whole POP3 client is implemented in this `POP3` class. We have a method run here which calls in order methods `startConnection`, `login`, `retrieveAllMessages` and `quit`.

2.3.1 startConnection()

Method `startConnection` creates instance of `Connection` class and then decides what type of connection should be created based on arguments.

- `openConnectionTLS` of Connection class is called in case that flag `-T` is specified
- `openConnectionSTLS` in case of the flag `-S`
- `openConnection` by default (no flags specified)

2.3.2 login()

This method handles the Authorization state of the POP3. The user is informed in case of an invalid username or password.

2.3.3 retrieveAllMessages()

This method goes through all messages and calls method `retrieveMessage` on them. If no flags are specified, this causes all messages to be downloaded; in the case of the `-n` flag, only new messages are downloaded, and in the case of the `-d` flag, downloaded messages are marked for deletion on the server. Saving of the messages to the file is handled by `FileManipulator` class.

2.3.4 quit()

A simple method that sends the POP3 `QUIT` command and then closes the connection.

2.4 Connection class

Class responsible for communicating with the POP3 server. Communication is done through the `BIO` object.

This class contains methods for opening a connection to the server. There are methods for TLS connection, unsecured connection, an unsecured connection that will use STARTTLS to promote to secure one. There is also a method for closing said connections - freeing memory and destroying objects used for this communication.

Two essential methods that allow the POP3 class to send commands to the server are `read` and `write`. These methods were taken from OpenSSL IBM documentation and are allowing us to send commands and read responses from the POP3 server

2.5 FileManipulator class

There are two methods used for file manipulation.

The first one, `saveMessage` is responsible for saving messages to the file; this method checks if the file already exists - if the file already exists, it checks `-n` flag to decide if the message should be saved again. The file name is requested from the second method of the `FileManipulator` class - `getMessageID`.

Method `getMessageID` attempts to get the value of `Message-ID` from the message header. If the value is not found error message is printed out on `stderr`. **This is a current major limitation of this client - it's not able to download messages that are missing-ID!**

3 How to use

To compile and run this program, you need the `openssl` library, `getopt` from `util-linux`, `g++` compiler, and `make`.

If you have all of the above, switch to the root project folder (the folder that contains `Makefile` and `main.cpp`), and you can run

```
$ make
```

This will compile the program and makes it ready to run. Object files and a copy of the executable is saved in the `build` directory.

3.1 Usage

```
$ ./popcl <server> [-p <port>] [-T|-S [-c <certfile>] [-C <certaddr>]]
                   [-d] [-n] -a <auth_file> -o <out_dir>
```

3.1.1 Required arguments

- `<server>` - POP3 server's domain name or IP
- `-a <auth_file>` - Authorization file - file containing username and password
- `-o <out_dir>` - Output directory - where the mails should be saved

3.1.2 Optional arguments

- `-p <port>` - Port over which we want to communicate with the POP3 server
- `-T` - Use secure connection from the start
- `-S` - Initiate unsecure connection and then use TLS to switch to secure one
- `-c <certfile>` - Path to the file that contains certificates
- `-C <certaddr>` - Path to the directory that contains certfiles
- `-d` - Delete downloaded messages
- `-n` - Download only new messages

3.2 Source files for the project

main.cpp,
pop3client/pop3.cpp,
pop3client/argumentParser.hpp,
pop3client/argumentParser.cpp,
pop3client/connection.hpp,
pop3client/connection.cpp,
pop3client/fileManipulator.hpp,
pop3client/fileManipulator.cpp,
pop3client/help.hpp,
pop3client/help.cpp
pop3client/returnCodes.hpp

4 Sources and literature

[2] ‘‘RFC1939.’’ Document Search and Retrieval Page,
https://datatracker.ietf.org/doc/html/rfc1939.
[3] ‘‘RFC2595.’’ Document Search and Retrieval Page,
https://datatracker.ietf.org/doc/html/rfc2595.
[4] ‘‘RFC5322.’’ Document Search and Retrieval Page,
https://datatracker.ietf.org/doc/html/rfc5322.
[5] ‘‘Secure Programming with the Openssl API.’’ IBM Developer,
https://developer.ibm.com/tutorials/l-openssl/.