VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

POP3 SERVER

ISA 2017

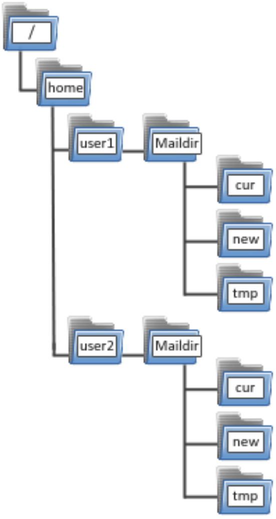
AUTHOR: Jakub Svoboda

BRNO 2017

1 Analysis

This documentation describes the implementation of POP3 server. Post Office Protocol is an application layer Internet standard protocol used by email clients to manage emails on a remote server. Throughout its development the Post Office Protocol has evolved over multiple versions, with version 3 being the most recent.

The implementation in this project contains all of the required, as well as the recommended functionality of the POP3 standard. It must also be capable of handling multiple users' concurrent connections and enable them to download emails from the server. The emails are saved in the **IMF** [4] format and are stored in a **Maildir** [2] directory (see image #1). The program operates with a single email directory accessed by a single user at each given time.



[img 1] The Maildir directory structure.

2 Implementation

The validity of the parameters that are given by the user is checked in the *validateParameters()* function. In there the arguments are parsed by the *getopt()* function. The program accepts a total of 7 arguments, with 3 of the being required, unless a single reset (-r) argument is passed. All the arguments' values are the stored in a Parameters structure for easier manipulation.

2.1 Authentication

After the parameters' values are saved, the program sets signal guards for interrupt signals and opens a semaphore. This semaphore is locked when a user logs in and has access to the Maildir folder. After this the server opens a connection using a non-blocking socket on the port specified by the user and waits for users to connect.

When a connection is established the server forks a separate process for the user to operate in. The parent process then waits for the next user to connect. Meanwhile the child process enables user to authenticate. If the server runs in a cleartext mode, the user logs in with the **USER** and **PASS** commands. If a safer method of sending the password is required, the user must use the **APOP** command to authenticate. The APOP method uses a md5 hash of the password and a timestamp, that is sent to the user in the welcome message, after the connection to the server is established.

2.2 Transaction

When a user has successfully authenticated and has access to the Maildir folder, he enters a Transaction. First, the server moves all new emails from the Maildir/new folder into the Maildir/cur folder. This is the only time (with the exception of TOP and RETR commands) when a server can read the emails. The size of the emails in octets is noted down by the server and the emails are assigned a number starting from one.

In the transaction state, the user has access to multiple new commands. **LIST** lists all emails with their size. **STAT** calculates the total size of all emails. The **RETR** command allow users to download email specified by its number. **DEL** allow user to delete emails. As the rcf [1] specifies, the emails are not deleted immediately after the command is sent, but rather after the user logs off. With this feature the user can undelete emails with a **RSET** command. The **UIDL** command lists the unique IDs of user's emails. The **NOOP** command does nothing. It merely replies with a positive response.

After the user is finished manipulating with emails, he/she signs off using the **QUIT** command, which ends the transaction phase and starts the update phase.

When the user has been inactive for a for at least 10 minutes, he is automatically logged of. When this happens, the connection is closed without getting to the update stage.

2.3 Update

The operations in the update state are run only if the QUIT command is executed from the transaction state. The user can also use the QUIT command in the authentication phase, but doing so will result in mere disconnect.

In the update state, the emails that are marked for deletion are deleted. After the files are deleted, the server closes the socket and the process handling the user is ended.

3 Usage

The server can be run in three modes. If only the –r parameter is passed, the server loads all file changes and resets them. Then all the log files are deleted and the server exits.

When the –h parameter is the only one specified, the help information is printed to the users terminal: `./popser [-h] [-a PATH] [-c] [-p PORT] [-d PATH] [-r]`

In all other cases, all the required parameters are checked. If all are present and the passed values are correct, the server starts and waits for user to connect. If all of the a required parameters are passed in addition to the –r parameter, the server restarts.

4 Extensions

This POP3 server implements the optional **TOP** command as well. When this command is executed in the transaction stage, the server returns *n* first line of requested email. The lines are counted from the beginning of email's body. The header is also sent to the user. When the number of lines requested is larger than the number of lines in the specified email, the whole email is sent to the user. Also the number of lines must be larger than zero and the requested email's number must be valid. If the email's number is valid, but it is marked for deletion, the user receives a negative response.

5 Code metrics

Lines of code: 720

Number of functions: 23

Size of the binary file: 276KB

6 Literature

1. https://tools.ietf.org/html/rfc1939
2. https://cr.yp.to/proto/maildir.html
3. https://tools.ietf.org/html/rfc2119
4. https://tools.ietf.org/html/rfc5322

Img 1: https://upload.wikimedia.org/wikipedia/commons/4/44/Maildir.png Licensed under the GNU Free Documentation Licence