Lab 2: TCP POP3 Server

RFC 1939 (https://tools.ietf.org/html/rfc1939) describes the Post Office Protocol - Version 3 (*POP3*). POP3 is used to allow a client to retrieve mail that a server is holding for it, in a much simpler way than the IMAP4 protocol. Most mailer applications (s.t. Thunderbird, for example) implement a POP3 client, that can be used to retrieve the mails from a server, that are then displayed by the mailer application.

In this lab, the thunderbird mail client (mailer) will be used to access the POP3 server you will develop. An SMTP server is provided to be able to send mails from the mailer. The SMTP server writes mails in a directory having the name of the mail destination user name. Your POP3 server will read mails from a directory having your user name.

1. First test the thunderbird mail client:

- start the provided SMTP server
- start a netcat server to replace the POP3 server you will have to develop later: `nc -v -l -p <port>`
- start thunderbird (see the command below) and configure an email account that works with the above SMTP and POP3 servers: this step will fail after the first exchanges with the POP3 server, of course.

Nb:

- to be able to launch several instances of thunderbird having different configurations (for each user), use the command: `thunderbird -no-remote -P <profile_name>`
- after having configured the email account, go to the account settings/copies & folders and uncheck "place a copy in:" ("when sending messages")

2. Then write a POP3 server compliant with the RFC 1939. This server should be launched with a directory name as argument. This directory will contain a subdirectory for each user having the same name as the user name, each containing the email files for the user (one file per email). In a "normal" mail server, this directory should be filled in by a SMTP server (cf https://tools.ietf.org/html/rfc5321). For this lab, email directories and files will be added "manually" to this directory.

This server should display all commands received and responses sent on the console for debugging purposes.

This server should be able to manage several clients simultaneously. The optional operations do not have to be implemented, except for the USER and PASS operations.

Technical instructions: choose between the two different versions (develop both if you have time):

- the "classical" version, with the `java.net` and `java.io` packages (using the `ServerSocket`, `Socket` and `FileReader` classes)
- the "optimized" version, using Java NIO (preferable, for efficiency reasons), with:
  - the `ServerSocketChannel`, `SocketChannel` and `Selector` classes for the POP3 server
  - the `Files#lines` method to access the lines of a mail contained in a file

Assignment: give an executable jar file with the Java sources. The program should display a help message describing the expected command line arguments when it is launched with the -h option. The server jar file should not contain the provided classes, but should run without error when the `io-utils.jar` is in the same directory as the server executable jar file.