Project

ETH Zurich

Hand-out: 25.2.2014
Team registration: 4.3.2014
Intermediate deadline: 25.3.2014
Final deadline: 6.5.2014
Project assistant: Mischael Schill (mischael.schill@inf.ethz.ch)
Mailing list: se-ccc@lists.inf.ethz.ch

1 Synopsis

The goal of this project is to apply the principles of SCOOP to a real world example: an Internet Message Access Protocol (IMAP) mail delivery agent (MDA).

2 Requirements

You must implement an IMAP4rev1 server according to RFC 3501 [1] using Eiffel/SCOOP. However, since properly handling Transport Layer Security (TLS) is difficult and not in the scope of this course, the required STARTTLS capability stated in section 6.1.1 of RFC 3501 is considered optional. The server does not need to support IPv6.

In addition, the server needs to be configurable using a configuration file. The configuration uses the syntax of an ini-file [4]. Since the server is developed and ran on a Linux system, use just Line Feed (LF, in Eiffel `%N`) as a line delimiter. An ini-file consists of four different kinds of lines:

1. Lines containing only whitespace (space and tab), which are ignored
2. Lines starting with a semicolon (`;`) for comments, which are ignored
3. Lines with a key-value pair delimited with an equality sign (`=`)
4. Lines solely containing a bracketed identifier, for example `[main]`, starting a section

2.1 Configuration

While the configuration file can be expanded as needed, it is mandatory for the server to interpret at least the following items:

Section `network`

- `address` (The address on which the server listens, default is 0.0.0.0, and represents all available IPv4 addresses)
- `port` (The port on which the server listens, default is 143)
- `workers` (The number of concurrent connections, default is 8)

Section `storage`

- `users` (The path to the file containing the user database, default is "users.csv")
- `mail` (The path to the root directory containing all the mail, default is "mailboxes")

Section `misc`

- `postman` (The username of the postman, who has writing access to all the inboxes of all users in order to post mail, default is having no postman)

2.1.1 Location of the configuration file

The program must implement a command line option `-c configurationfile` which specifies the location of the configuration file.

2.1.2 Example configuration file

```
[network]
;Bind to localhost only
address=127.0.0.1
;Using port 1143 for running server as non-root user
port=1143
workers=16
[storage]
users=/home/ccc/Project/Sources/etc/users
mail=/home/ccc/Project/Sources/var/mail
[misc]
;postman does not work yet
;postman=postman
```

2.2 Users

For this project, usernames and passwords may only consist of alphanumeric characters. The user database file is a comma-separated file with usernames and passwords. The special user denoted as "postman" in the configuration file is to be treated differently from other users. The postman sees all inboxes of all other users, eached named accordingly. This enables a program to deliver mails to the inboxes of users. The postman user should not be able to see the actual mails and mailboxes other than the inboxes.

2.2.1 Example user database

```
joe,joe52
jane,ImLovinIt
postman,ThereWillBePost
```

3 Environment

A VirtualBox image is provided on the course website [2] that has the necessary tools preinstalled. Both username and password are set to ccc by default. The root password is also set to ccc.

3.1 Setting up the virtual machine

1. Install VirtualBox from [3]
2. Download the Virtual Machine from `http://se.inf.ethz.ch/courses/2014a_spring/ccc/vm/CCC.zip`
3. Unpack the Virtual Machine
4. Start Virtual Box
5. Add the VM through Machine → Add ...
6. Start the virtual machine
7. Log in with username and password ccc
8. Start a terminal and type `setup.sh`
9. The VM is now set up for working on the project

3.2 Home directory structure

Desktop Contains the files displayed on the Desktop

Project The root directory of the project

- **Sources** The working copy of the team's Subversion repository
  - **scoopmda** The EiffelStudio project directory
  - **report** Where the report should be added (LaTeX preferred, otherwise PDF)
- **Libraries** Contains the provided libraries
  - **net2** The networking and file library
  - **estring** Expanded strings
  - **nspr** Support for the Netscape Portable Runtime (used by net2).

Workspace Contains two pre-configured EiffelStudio SCOOP projects that can be used for exercises

3.3 Snapshots

The system is configured to make hourly snapshots of the home directory. It retains the last 24 snapshots. The snapshots are available from `/home/.snapshot`. **Do not rely solely on these snapshots as your backup!**

3.4 Important programs

The task bar contains five icons:

1. Terminology – a terminal
2. EiffelStudio
3. a shortcut for updating the libraries
4. rapidSVN – a graphical Subversion client
5. TexMaker – a LaTeX editor

3.4.1 netcat

netcat is a useful command line (terminal) tool to test the server. Calling `nc -C 127.0.0.1 1143` establishes a TCP connection to port 1143 of localhost. The `-C` option tells netcat to use Carriage Return/Line Feed (CRLF) for ending a line, which is how messages are delimited in IMAP according to the specification. Use multiple terminals for multiple instances of netcat when testing multiple concurrent connections.

3.4.2 ImapTester

A small Java program to test the imap server is available at `http://se.inf.ethz.ch/courses/2014a_spring/ccc/ImapTester.zip`. It can be run either on the host or inside the VM. It currently only supports the commands for the first Milstone, and will probably be expanded later. The zip file contains a NetBeans project, so the program can be modified. The dist directory contains a jar file with the compiled program.

3.5 Installing additional programs

The hard disk space for programs is limited to keep the VM file small. A full installation of TeXlive is therefore not possible. If you want a VM that has LaTeX installed, ask the project assistant for a variation of the VM with more space for programs and an installed version of LaTeX. The free space for the home directory should be large enough.

3.6 Firewall

The firewall is configured to allow access to the ports 143, 1143 and 1144. Since port 143 can only be bound by root, it is recommended to use 1143. Secure Shell (SSH) is both disabled and blocked since the password is too simple to guess. Enabling of SSH is only recommended if both the password of ccc and root is changed first. The VM is configured to forward these ports from the local machine to the VM if you want to test the server from outside of the VM.

3.7 VM configuration

The VM is configured for 512MB of RAM and one core. If your computer has more cores and/or more memory, you might want to change these parameters to speed up the VM.

3.8 Networking library

The networking library used for the project is quite new. If something does not work as expected, report it to the course mailing list. Make regular use of the `Update` icon on the task bar to keep the library updated.

4 Deliverables

Delivery is done through the team's subversion repository. By default, the last commit before the deadline is considered. If a previous commit should be considered, please send a message to the project assistant.

4.1 Report

It is expected that the program is well documented in the report. Up to three points can be deducted for poor readability, structure or grammar. The report values up to 15 points and needs to encompass the following:

- Comprehensive user manual (3 points)
- Architecture description and the design decisions (5 points)
- Describe how SCOOP is used and how it helped or hindered the implementation (3 points)
- Describe your test cases and their coverage (4 points)

4.2 Program

It is expected that the program code is readable and documented. Poorly written code hampers the assistants' ability to award points if the program does not work as expected.

4.3 Milestones

4.3.1 Team registration – Deadline: 4.3.

Register your team by sending a message to the project assistant.

4.3.2 Basics (5 points) – Deadline: 25.3.

The basic version of the program has to read the configuration and user files and support a single connection. Points are awarded as follows:

- Supported commands: `NOOP`, `CAPABILITY`, `LOGOUT`, `LOGIN` (4 points)
- Test cases (1 point)

4.3.3 Final submission (45 points) – Deadline: 6.5.

The final program has to support (up to 8) connections from multiple concurrent users according to the `workers` configuration property. Points for this milestone are awarded as follows:

- Implementation of all commands except for `STARTTLS` (20 points)
- Parallel execution of compatible commands (5 points)
- Test cases for all commands (5 points)
- Report (detailed in section 4.1) (15 points)

5 Teams

You can work in teams of up to three persons. In order to register your team, send a message to the project assistant by 4.3.2014.

6 Repository

Every team has access to its part of the Subversion repository of the course. You are free to use it for development, and we encourage you to frequently commit your changes in case your computer breaks down. The repository is st up by calling `setup.sh`, as described in the step-by-step guide.

The URL of the repository is `https://svn.inf.ethz.ch/svn/meyer/ccc/trunk/teams/teamname`, where `teamname` stands for the name of your team.

7 Support

For help, consider using the course mailing list. This list contains all students in the course in addition to the assistants.

There is also the possibility to ask questions during the office hours. Office hours are on Friday between 14:00 and 16:00 in RZ J6. Additional office hours are every Wednesday from 15:15 to 17:00 in RZ F21 if there is no seminar. Please drop a message to the project assistant at least 15 minutes in advance.

For resources on SCOOP and EiffelStudio, you can also consult the course website [2].

8 Updates

Please check the course website [2] for potential updates of the project description and supporting material.

9 Hints and known issues

9.1 Once routine

In order for a separate once routine to be executed only once for the whole system, the phrase `("PROCESS")` has to be added immediately after `once`.

9.2 Finalization and expanded types

There is currently a bug in EiffelStudio that corrupts expanded object returned from separate queries in finalized programs. Because of it, we do not require that the server works as a finalized executable.

9.3 ESTRING as arguments

The `ESTRING_*` classes are not standard in Eiffel. They are very convenient when using SCOOP and therefore used by the (new) networking library. Most of the time, they can be used as a drop-in replacement for regular `STRING`s. However, if a separate feature has an argument of type `ESTRING_8` or `ESTRING_32`, passing a constant (e.g. `"Hello World!"`) does not work. There are two workarounds: either assigning the constant to a local variable of appropriate type or using the `from_string_8` constructor of `ESTRING_*`

References

[1] M. Crispin. INTERNET MESSAGE ACCESS PROTOCOL. RFC 3501. Internet Engineering Task Force. March 2003. Available at http://tools.ietf.org/html/rfc3501.

[2] CCC. ETH Zürich. http://se.inf.ethz.ch/courses/2014a_spring/ccc/, 2014.

[3] VirtualBox. Oracle. https://www.virtualbox.org/, 2014.

[4] INI file. Wikipedia. https://en.wikipedia.org/wiki/INI_file, 2014.