VE572 — Methods and tools for big data

Lab 1

Jing and Manuel — UM-JI (Summer 2018)

Goals of the lab

- Java programming
- Network socket programming
- Documentation searching and reading

Teaching team's warning message.

While ve401 is a fitting prerequisite for the first part of the course, various topics not taught in JI must be known and mastered to fully succeed in ve572. Therefore this lab, offered in week 2, should allow students to self-evaluate their ability to survive in the second half of the course.

Topics assumed as known in the second part of this course but having no prerequisite:

- Advanced knowledge of the Linux operating system (partially covered in ve482)
- Java programming
- Basic network programming

Students overriding ve482 should be especially careful regarding the amount of knowledge that will be required in the second part of the course. Important topics covered in ve482:

- Basics on filesystems
- Scheduling
- Basics on distributed systems
- Basics on processes

Note: this lab is expected to be long and not easy. Do not wait the last minute to start, and closely adhere to the instructions. In particular do not forget to precisely evaluate and report the time spent to complete all the tasks.

1 Introduction

Beyond the initial warning message, the content of this lab is of a major importance for the rest of the course as most of the information learnt here will be reused at a later stage.

1.1 Lab outcome

The main goal of this lab is to improve the ability to work in an unfamiliar environment: most topics are likely to feel new for many students. Most important concepts encountered along the way:

- Search for, find, read, and understand industry specifications;
- Improve Java programming skills[1];
- Search, find, understand, and code using external Java libraries;
- Read libraries' licences and decide of their adequacy for the company;
- Find a VPS provider and setup a server;
- Improve socket programming skills;
- Get more comfortable with command line;

An ability matrix showing the basic abilities required along the lab is provided in appendix A.

[1] Refer to appendix C for a brief introduction to Java

1.2 Tasks

The core part of the lab consists in six main tasks:

1. Own a remote server on a public cloud platform.
2. Implement a socket protocol in Java.
3. Deploy a program on a remote server.
4. Write a Java program to extract data from an XML file.
5. Dump the data from a format specific binary file.
6. Complete all the tasks into a remote server application.

Warning: completing all the tasks represents a full week of work for a group of two to three people. Do not start late...

2 General setup

This section provides all the necessary details to complete the lab.

2.1 Background

As a data analyst working for the Hexagon, the Department of Harmony of the United Cities of Oceania, your are asked to investigate the meddling suspicion of a foreign power into the last elections. For your work you are provided with potential evidences collected from FaceDirectory Inc., a famous worldwide social network. To make sense of the data you are requested to prepare the following setup.

- Develop and deploy a server program that accepts, analyses, and returns the results of queries on data files.
- Follow the specifications of the server program and protocol as described in section 2.4.
- Deploy the server program on a remote server.
- Figure out how to parse and extract data from the data files.
- Freely create a team of no more than four data analysts.

The quality of your work will be assessed. While a low quality work can immediately put an end to your contract, in most cases it should have no bearing on your annual evaluation.[2]

2.2 Data package

The full data package you are receiving when you are assigned the task consists of (i) a `.pdf` file containing some specification as well as some less relevant information, (ii) two sets of data files, each composed of an `.xml` and a `.bin file`, (iii) the extracted content of one of the two sets of data files, and (iv) two SSH public keys.

The first data set with the extracted content is called *decoding* while the second one, which is to be analysed, is called *validating*.

[2] The lab is graded, but its grade will not count toward the final grade. However in certain cases we may reject the ve482 override application (cf. section 3.2).

Unfortunately the format of the data files is unknown. The only hints you are given is that a data file contains one or more data series, and that each series has a name and a few other attributes.

The goal is evidently to extract the data for the unknown data set.

2.3 Remote server

The program should be deployed on a remote server of your choice. Free services exist and can be used, e.g. Google Cloud Platform and GitHub's Educational package. Other alternatives, for instance commercial solutions, are also acceptable.

Once the server is setup, create the user ve572. Use the two SSH public keys found in the data package to allow the managers[3] to remotely connect to the server. This account should have administrative access to the server[4]; the method is left to your choice, but should be documented.

2.4 Specifications

The server program should accept a TCP connection used by the client to (i) send both the queries and data files, and (ii) receive the results. The protocol follows a client driven request - response approach.

Instructions that can be sent by the client:

- `BEGIN;` used by client to initiate a communication.
- `SIZE Type Size;` used by the client to send the size `Size` (in bytes) of the file of type `Type` to be sent on the server. The value `Type` can be either `XML` or `BIN`. The file is effectively sent as soon the server acknowledges the request, i.e. it replies `OK`.
- `QUERY Op Name;` used by the client to request the operation `Op` on the data series `Name`. The operation `Op` is one of `MAX`, `MIN`, `AVG`, `SUM`, and `MEDIAN`.
- `END;` used by the client to conclude the session. The session is effectively closed as soon as the server acknowledges the reception of the instruction.

Notes: all the text, sent or received, should use the UTF8 encoding; The protocol is case-insensitive and each textual client request ends with a semicolon "`;`"; Instructions are always expected in the following order: `BEGIN`, `SIZE XML`, `SIZE BIN`, a certain number of `QUERY`, and `END`; Files are sent in binary format;

Behaviour of the server:

- The server acknowledges every client request by replying `OK`.
- Upon receiving a query the server replies `OK` and then processes it. The result is sent back to the client in the format `RESULT Name OF Quantity Value Unit FROM Count POINTS`. The parameters appearing in the reply are
  - `Name`: the name of the data series;
  - `Quantity`: the name of the physical quantity (e.g. time, distance, pressure, etc.) defined in the XML file;
  - `Value`: the value determined by the server based on the request;
  - `Unit`: the unit of `Value`;
  - `Count`: the total number of data points in the analysed series;
- At the end of a session the server waits for new session from the same client or new ones.

[3] Teaching team

[4] We understand the server contains sensitive data and will not temper with any else that the files related to the lab.

Example

Discussion between a client and a server to send a set of data files.

```
Client: BEGIN;
Server: OK
Client: SIZE XML 512;
Server: OK
The client sends the 512 bytes XML file.
Server: OK
Client: SIZE BIN 1024;
Server: OK
The client sends the 1024 bytes binary file.
Server: OK
Client: QUERY MAX startTime;
Server: OK
Server: RESULT startTime OF Time 75 ms FROM 42 POINTS
Client: QUERY AVG totalLength;
Server: OK
Server: RESULT totalLength OF Length 12.3 meter FROM 7 POINTS
Client: END;
Server: OK
The client closes the connection.
```

3 Evaluation

The overall evaluation of the competence for the lab splits into two stages: the lab session and the final submission.

3.1 Lab session

The actual lab session will be handled as a research meeting where the groups present their research advances. A presentation should

- Not exceed 3 min.
- As much as possible, feature technical demonstrations rather than slides.
- Focus on a single task: precisely expose the problems, the techniques that failed or worked, etc.
- Be different for each group. Even if the task is the same, the content must vary, e.g. present an alternative strategy to overcome a problem that was previously discussed by another group.

Each presentation will be followed by a five minutes question and answer session. Everybody **should participate** to the discussion such as to improve the quality of the meeting and benefit from each others' research and work.

A progress evaluation sheet will be issued before the lab session. Each group should fill it in such that other group know what questions can be directed to them.

Warning: although not directly graded, this stage can significantly influence the judgement of the teaching team on your level of preparation for this course.

3.2 Final submission

Subsequently to the research meeting each group should work further on the lab and submit both the final version of the code and report at the beginning of the following lecture. Although no precise guidelines on the format of the submission are enforced the report should

- Provide all the necessary information to access the server (e.g. IP address, way to gain administrative rights, etc.)
- Contain the documentation of the code.
- Detail the progress on each task.
- Emphasize the main discoveries and point learnt in the course of the lab.
- Honestly present the failures and the attempts to overcome them, as well as simple explanations on why they did not work.
- Provide a self-evaluation for each task. An example can be found in appendix B. The ability matrix (appendix A) may help in the self-valuation process.

The final letter grade for this lab will consist of six letter sub-grades, ranging from A to D, no partial letter grade, and apportioned as follows.

1. Own a remote server on a public cloud platform: 10%.
2. Implement a socket protocol in Java: 25%.*
3. Deploy a program on a remote server: 15%.#
4. Write a Java program to extract data from an XML file: 25%.
5. Dump the data from a binary file: 15%.†
6. Complete all the tasks into a remote server application: 10%.

* For this task only the socket protocol part will be tested. Having it working locally is sufficient to be awarded the full mark.

# For this task a fully running program on the remote server in required.

† If this task is completed before the lab session, email the dumped data along with the structure of both the XML and binary files of the validating data set to the teaching team. Provide simple explanations on the external resources you have used. If the results are correct the group automatically achieves full mark for this task.

Important note: if task 5 is completed before the lab session then the program does not necessarily need to work on any data file. Otherwise the program should dump the data from an arbitrary data file to achieve a C.

Warning: an eligible grade for this lab is **no less than C on each task with a final grade at least equal to B**. Groups unable to perform better should carefully consider their enrollment in the course. Besides, students overriding ve482 will see their **application rejected** if they are unable to comply with the above grade requirements. This lab containing much basic knowledge onto which the second half of the course relies, students without a satisfying grade on this lab will **not be allowed to carry on the future works** until they submit a fully working program.

A Ability matrix

An overview of the basic abilities required by the various tasks in shown below.

Abilities	Tasks					
	1	2	3	4	5	6
Efficiently search and use information found on Internet	√		√	√	√	√
Program and use libraries in Java		√		√		√
Understand basics of networks and socket programming		√				√
Work with the Linux OS on a remote server	√		√			√
Perform independent study and research					√	√
Dauntlessly face uncertainty with outstanding bravery	√	√	√	√	√	√

B Report sheet

The goal of the self-evaluation is to estimate the average time spent and difficulty perceived over this lab and help students realising how much work will be required from them in the second part of the course. This is especially important to students overriding the ve482 prerequisite.

Please be honest and accurately report your evaluation. There will be no reward for "fast students" or penalties for "slow ones". We only expect it to help everybody evaluating their ability to survive in this course before the dropping period ends.

Reminder: the teaching team will only check the completeness and quality of the work. Students cheating during the lab or on the time they spent and the difficulty perceived will bear the consequences of their choice...

Tasks	Time (hh:mm)	Difficulty level				
		1	2	3	4	5
1. Own a remote server on a public cloud platform		○	○	○	○	○
2. Implement a socket protocol using Java		○	○	○	○	○
3. Deploy your socket program on your remote server		○	○	○	○	○
4. Write a Java program to extract data from an XML file		○	○	○	○	○
5. Dump the data from a binary file		○	○	○	○	○
6. Complete all the tasks into a remote server application		○	○	○	○	○

C A C++ Hitchhiker's Guide to the Javanian Galaxy

Java is a compiled language, yet it runs on the Java Virtual Machine (JVM).
Java programs need to be compiled and run on a runtime called Java Runtime Environment (JRE). The Java development environment requires a package called Java Development Kit (JDK).

Java is fully object-oriented.
Java programs are organized in terms of classes. All variables or functions must reside within the scope of a class. All classes always inherit a common base class "Object".

The Java memory management features a garbage collector.
Object are automatically deleted and recycled when no longer needed. However, in some cases, you have to pay a significant performance price for that feature. Stay alert.

Java is statically and weakly typed.
Like C++, all Java variables are typed, and their type must be known at compile time. If necessary variables can be cast.

Java objects are always passed by reference.
Java contains a few built-in types that are passed by value. However all objects are passed by value-references. Think of them as pointers that automatically dereferences themselves.

Java is polymorphic by default.
By default, all methods in a class are "virtual". Subclasses always override base class objects.

Java allows only single Inheritance and features Interfaces.
Only single inheritances are allowed. Interfaces in Java are similar to pure abstract base classes in C++.

Generics are done by type-erasure.
Java also supports Generic containers, similar to `std::vector<>` or `std::list<>` in C++. However, unlike C++, Java uses type-erasure to handle generics. In contrast, C++ uses type specialization.

Java supports reflection and introspection.
It is possible to ask a class to print out its supported methods, or look up a class by string, at runtime.

Have fun with Java.
Yep. Enjoy Coding!