Exercise Session 1

Getting Started with Emerald

Oleks Shturmov

<olekss@uio.no> / <oleks@oleks.info>

University of Oslo

IN[59]570: Distributed Objects

January 28, 2019

The source code for these slides is maintained here:

https://github.com/emerald/in5570v19/tree/master/exercise-sessions/01

Agenda

1. Some Bits About Me
2. Elements of Programming in This Course
3. What Is An Emerald Node?
4. How Will You Run Emerald?
5. Survey
6. Get Some Emerald Nodes Up and Running
   - Install, Compile, and Run an Emerald program
7. In Emerald, What Is a Distributed System?
8. What Next?

Some Bits About Me

- Born in Kyiv, Ukraine in 1990
- Parents immigrated to Denmark in 2001
- Hence, fluent in Ukrainian, Russian, English, and Danish
- BSc and MSc in CS at the University of Copenhagen (DIKU)
- Teaching and research at DIKU before coming to UiO
  - Operating systems
  - Programming languages
  - Computer architecture
- Got to know Eric while teaching at DIKU
- Started on a PhD under Eric in December 2017
- Topic: Programming Heterogeneous, Distributed Systems

Elements of Programming in This Course

1. Writing **programs**
   - Submit readable, preferably working code
   - Test your code, and tell us how to reproduce your test results
   - Refactor your code once it works, and before you submit
2. Writing programs in a **text-based** programming language
   - Use indentation to indicate program structure
   - Use adequate naming
   - Organize code into methods and classes
   - ~~Organize code into files and directories~~ (Maybe later)
   - Apply other common elements of (text-based) programming style
3. Writing programs for **distributed** execution
   - Program fragments execute concurrently on (distant) nodes
   - Program fragments coordinate to get common tasks done
   - Nodes are unreliable (the software/hardware beneath you may fail)
   - Node-to-node communication is unreliable

What Is An Emerald Node?

In the context of this course, an Emerald constitutes the following:

1. Some general-purpose network-enabled hardware (laptop or server)
2. Running some general-purpose operating system (typically, Linux)
3. Having some associated IP address or hostname (e.g., `localhost`)
4. Running an (old and dusty) Emerald virtual machine at a designated port

Hence, an Emerald node is uniquely identified by:

`<host>:<port>`

where

- `<host>` is an IP address or hostname, and
- `<port>` is an integer between 0 and 65535.

How Will You Run Emerald?

1. Within a Docker container (recommended), or natively
   - For basic development only
   - Docker is recommended to ensure a consistent experience
2. Within a virtual machine we hand out (e.g., using VirtualBox)
   - For advanced development, using network emulation for tests
3. On a number of machines in a world-wide network (PlanetLab)
   - For a real-world (latency) experience

Survey (1/2)

- Number of participants in the survey: 17 (including Oleks)
- Number of PhD students: 1 (Oleks)
- Number of external students: 1
- What operating system are you running?

Linux	**MacOS**	**BSD**	**Windows**	**Other**
7	5	0	6	0

- What is your favourite general-purpose text-editor? (Which you also use.)

Vim	**emacs**	**Atom**	**Notepad++**	**VS Code**	**Sublime**
2	4	7	1	3	2

- What is your favourite general-purpose IDE?

Eclipse	**IntelliJ**
3	3

Survey (2/2)

- Have you written, executed, and tested a distributed program before?

Yes	No	Not sure
5	11	1

- Have you heard of Erlang?

Yes	No
7	10

- What programming languages do you know? (For example, you have written at least 10.000 lines-of-code, or have had extensive course work using the language.)
  - Java
  - C
  - C++
  - Python
  - JavaScript
  - Haskell

Get An Emerald Node Up and Running

- The Docker image prepared for this course is on Docker Hub:

  `https://hub.docker.com/r/portoleks/in5570v19/`

- Once you have Docker installed, you can run it as follows:

```
$ docker run -it --rm portoleks/in5570v19:latest
```

- This lands you in a BASH shell, having the following binaries:
  - ec (Emerald compiler)
  - emx (Emerald virual machine / execution engine)
- **NB!** The file-system is ephemeral; once you exit (type `exit`, or press Ctrl+D), the files you create here are lost!
- To run with **your working directory**[1] mounted, do this instead:

```
$ docker run -it --rm \
    --volume "$(pwd):/home/docker/src/" \
    --workdir "/home/docker/src/" \
    portoleks/in5570v19:latest
```

NB! In Windows PowerShell, use ‘ (backtick) instead of \ (backslash).

[1] Make sure you are in a project directory before you run this command

Print Hello, World!

Here is a program with some observable behaviour:

```
const main <- object main
  initially
    stdout.putstring["Hello, World!\n"]
  end initially
end main
```

To compile and run:

```
$ ec hello.m    # Assuming you call the above file hello.m
$ emx hello.x   # Assuming ec went well, you'll get a hello.x
```

Print Hello, World! Everywhere (1/2)

This code asks every node to print `Hello, World!`
in each their own standard output stream:

```
const helloall <- object helloall
  initially
    const home : Node <- locate self
    const all <- home$activenodes
    var elem : NodeListElement
    var friend : Node
    for i : Integer <- 0 while i <= all.upperbound by i <- i + 1
      elem <- all[i]
      friend <- elem$thenode
      friend$stdout.putstring["Hello, World!\n"]
    end for
  end initially
end helloall
```

Print `Hello, World!` Everywhere (2/2)

To compile and run (on 3 nodes):

```
$ ip addr                       # Determine the IP address of master
...
... eth0...
  inet 172.17.0.2...            # Here it is, under eth0, inet
$ emx -R                        # Start an Emerald master node
Emerald listening on port 17099...
```

```
# Start another node
$ emx -R172.17.0.2:17099    # The port is (often) optional
```

```
# Start another node, and run helloall.x
$ ec helloall.m
$ emx -R172.17.0.2:17099 helloall.x
```

NB! Don't use a space between `-R` and the node identifier (e.g., write `-Rlocalhost`, **not** `-R localhost`).

In Emerald, What Is A Distributed System?

Object-orientation
In the Emerald world-view, a distributed system is a collection of communicating objects, distributed across a number of nodes.

Object mobility
Although an object is always instantiated on a particular node, it may move from node to node throughout its lifetime.

What Next?

- Write scripts to simplify the tasks we did today
- Start reading the below:

Raj, Tempero, Levy, Black, Hutchinson, and Jul (1991), Emerald: A general-purpose programming language. Software: Practice and Experience, Vol. 21, No. 1.

`https://www.uio.no/studier/emner/matnat/ifi/INF5510/v15/pensum/SPE-paper-1991.pdf`

Raj, Tempero, Levy, Black, Hutchinson, and Jul (1991), Technical Report: The Emerald Programming Language

`https://www.uio.no/studier/emner/matnat/ifi/INF5510/v15/pensum/Report.pdf`