Exercise Session 0

Getting Started with Emerald
Pre-session slides (to be updated after session)

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

University of Oslo
IN[59]570: Distributed Objects

January 20, 2021

The source code for these slides is maintained here:
https://github.com/emerald/in5570v21/tree/master/exercise-sessions/00

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 TCP/IP-enabled hardware (e.g., your laptop)
2. Running some general-purpose operating system (e.g., Linux, macOS)
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 Docker containers (recommended), or natively
   - For basic development
   - Docker is recommended to ensure a consistent experience
2. Within many Docker containers
   - For robustness testing (fault emulation)
   - Docker enables to emulation of network conditions on one machine
3. On a number of machines in a world-wide network (PlanetLab)
   - For a real-world (latency) experience

Survey (1/3)

I would like to get to know you, so please take my introductory survey by scanning the QR code, or clicking the link below. (Powered by the UiO installation of Office 365.)

https://forms.office.com/Pages/ResponsePage.aspx?id=EWg7RqSwKku5MnLEyXDF0jX8fpCu28JLoOqRVD4wrQRUN0lMTFBCRVRaNk5BWVMyWlUwS1ZVMUlLWC4u

Survey (2/3)

- Number of participants in the survey: ?
- Number of PhD students: ?
- Number of external students: ?
- What operating system are you running?

Year	Linux	MacOS	BSD	Windows	Other
2019	7	5	0	6	0
2020	6	5	0	7	0
2021	?	?	?	?	?

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

Year	Vim	emacs	Atom	Notepad++	VS Code	Sublime
2019	2	4	7	1	3	2
2020	1	0	7	1	8	2
2021	?	?	?	?	?	?

Survey (3/3)

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

Year	Yes	No	Not sure
2019	5	11	1
2020	4	15	0
2021	?	?	?

- Have you heard of the Erlang programming language?

Year	Yes	No
2019	7	10
2020	8	11
2021	?	?

- 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
  - SML
  - Scheme

Get An Emerald Node Up and Running

- First, install Docker
- Once you have Docker installed, you can run our image as follows:

```
$ docker run -it --rm portoleks/in5570v21:latest
```

- **NB!** Windows users see also next slide
- 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/in5570v21:latest
```

- For more on what this does see the README at `https://hub.docker.com/r/portoleks/in5570v21/`

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

Windows Users

- If you are using Windows 10 Home Edition (or similar):
  - Download and install Docker Toolbox: `https://docs.docker.com/toolbox/overview/`
  - Make sure your folder with Emerald code (from where you also execute the `docker run` command), is under `C:\Users`
  - Make sure that folder is not marked read-only (placing it under your user directory should work)
- If you are using Git BASH, you should be OK with the above command.
- If you are using PowerShell, use this command instead:

```
$ docker run -it --rm `
    --volume "${PWD}:/home/docker/src/" `
    --workdir "/home/docker/src/" `
    portoleks/in5570v21:latest
```

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.

Declare A Sieve Object

Problem presented in Lecture F1:
`https://www.uio.no/studier/emner/matnat/ifi/IN5570/v21/lecture-slides-v21/`

Eric's (Danish) formulation: Et objekt som accepterer det første tal, som det får, herefter vil det videresende til en kopi af sig selv, de tal, der IKKE har det første tal som divisor. Send tallene fra 2 til 100 igennem sien!

My humble translation: An object that takes in a number, and forwards to copies of itself, those numbers, that DO NOT have the given number as a divisor. Sieve only the numbers from 2 to 100!

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`