Language Study: Erlang

CMPT 333

Lab 1 - 60 points

Goals

- to learn about Erlang's history
- to become familiar with the Erlang interactive environment and get comfortable working in there by playing around with assorted stuff.

Requirements and Notes

Part One

- Read chapters 1, 2, and 3 in our book.
- Read the papers *A History of Erlang* and *The Development of Erlang* by Joe Armstrong. They are linked on our class site.
- Listen to Joe Armstrong being interviewed on *episode 89 of Software Engineering Radio*. It's also linked from our class site.

Everything from these sources is fair game for tests so sit back, relax, and dig in to the history and philosophy of Erlang.

Part Two

- Download and install (or build from its source code) Erlang on your own system.
- Fiddle around in the interactive environment by working through all of the examples in chapters 2.1 and 2.2 as well as 3.1 through 3.4 in our book.

Part Three

- In a new LaTeX document, compose your answers to the following Erlang-related questions:
  1. What is single assignment?
  2. What's the difference between a *bound* and *unbound* variable?
  3. How does variable scope work in the Erlang environment?
  4. Does Erlang implement mutable or immutable memory state? Why?
  5. Describe Erlang's memory management system.
  6. What does "Erlang" mean or stand for, if anything?
  7. Contrast "soft real time" from "hard real time".
  8. Why is Erlang so well suited for concurrency-oriented programming?
  9. Explain Erlang's "let it crash" philosophy.
  10. What's the difference between a tuple and a list?
  11. What's BEAM?
  12. How can it be that we can create more Erlang "processes" than are allowed for in the operating system?

Resources

- Our book, links on our class website, and Erlang itself.

Hints

This is still pretty easy. Show off what you've learned about LaTeX and Git.

Submitting Your Work

Compose your answers to Part Three in LaTeX and commit your final PDF and LaTeX source file(s) to the *Lab 1* directory in your **private** GitHub repository on or before the due date (see our syllabus).