

Assignment 3: Io

Assigned: Friday, 4 February 2011

Due: 11:00 p.m., Thursday, 10 February 2011

Summary: In this assignment, you will further explore some of the key ideas from the reading on Io.

Purposes: To give you more experience with Io and its underlying concepts, particularly

Expected Time: Three to four hours.

Collaboration: I would prefer that you work on this assignment in groups of two or three. You may also work on this assignment alone or in groups of up to size four. You may consult with anyone you wish on this assignment (including other classmates), provided you cite your consultants.

Submitting: Upload a tarball to Pioneerweb. See further details below.

Warning: So that this assignment is a learning experience for everyone, I may spend class time publicly critiquing your work.

Assignment

1. Tree Nodes

Implement a node “class” that has slots for the node’s data and the node’s children. Your class should also support an `addChild` method.

2. Traversing Trees

Using Io’s actors, write a tree printing algorithm using the following basic strategy:

```
To print a node
  If the node has already been printed, stop
  Otherwise,
    Print the node
    Mark the node as printed
    Schedule all children to be printed
```

For “fun”, add a random delay to the printing strategy.

3. Modeling Expressions

As you no doubt know, we can model expressions as trees.

- A constant is an expression.
- A variable is an expression.
- An operator applied to two expressions is an expression.

Implement this model and a simple evaluation procedure for arithmetic expressions in Io.

E.g., I might create the expression $x + (3 * 4)$ with

```
Expression application(  
    "+",  
    Expression variable("x"),  
    Expression application("*",  
        Expression constant("3"),  
        Expression constant("4")))
```

or

```
Application new("+",  
    Variable new("x"),  
    Application new("*",  
        Constant new("3"),  
        Constant new("4")))
```

4. Simulating Distributed Computing

In one model of distributed computing, the system maintains a list of tasks to compute. Individual computing nodes take turns grabbing tasks from the list, computing with those tasks, and adding subtasks on to the list. A shared data store keeps track of the results of tasks.

For example, if a computing node is called upon to compute the result of $(3*4) + (5-6)$ and store it in slot `s0`, it might (a) create two new data slots, `s1` and `s2`, (b) create a task to compute $3*4$ and store the result in slot `s1`, (c) create a task to compute $5-6$ and store the result in slot `s2`, and (d) create a task to compute `s1 - s2` and store the result in slot `s0`.

Develop a simulation of this computing model for the expressions you developed in the previous exercise. You'll have two "shared" objects that represent the list and data store. You'll create a `ComputeNode` class that grabs tasks and builds new tasks. Then you'll build a simple task and a set of nodes and put them to work.

Important Evaluation Criteria

I will look at the standard "C" criteria: Is your code **correct**? Is your code **concise**? (This second attribute is particularly appropriate in Ruby.) Is your code appropriately **commented**? Have you used any particularly clever techniques. How **comely** is your code?

Submitting Your Homework

Create a directory for this assignment of the form `hw03.names`. Within that directory, create subdirectories for each part of the assignment (e.g., `part1`, `part2`). Each subdirectory should contain:

- The code that you have written for that part of the assignment.
- A typescript or typescripts in which you demonstrate the utility of the code you have written.
- (Optional) Unit tests.

- (Optional) A README file with any additional info you want to provide.

Make a gzipped tarball of the assignment directory. (If you don't know how to do that, let me know.)
Upload it via the Assignment 2 link on Pioneerweb.