Programming Languages as Operating Systems

Matthew Flatt

University of Utah
Programming in Heaven

\[ \lambda \]
Programming in Heaven
Programming in Heaven

language run-time

101010
010101
Multi-Programming
Multi-Programming
Multi-Programming

\[ \lambda \rightarrow 1010101010 \rightarrow \lambda \]

\[ \begin{array}{c}
101010 \\
010101
\end{array} \]
Multi-Programming in Heaven

\[ \lambda \to (\lambda x \ldots) \to \lambda \]

\[
\begin{array}{c}
\text{101010} \\
\text{010101}
\end{array}
\]
Multi-Programming in Heaven

(\lambda (x) \ldots)

language as OS
Languages as Operating Systems

Language as OS $\Rightarrow$ process controls:

- Separate threads of evaluation
- Separate process-specific state (e.g., current directory)
- Separate graphical event loops
- Ability to terminate a process and reclaim its resources

Language-based OSes: Pilot [Redell80], SPIN [Bershad95], ...

Extended languages: JKernel [Hawblitzel98], Alta [Tullman99], KaffeOS [Back00], ...
Example: Processes in a Language

DrScheme

user's program

Run DrScheme
More Process Examples

The image shows an installation process with the following steps:

- Updating `bin/icfp-robot-client`
- Updating `bin/simple-icfp-robot`
- Updating `bin/setup-plt`
- Updating `bin/framework-test-engine`
- Updating `bin/framework-test`
- Updating `bin/web-server`
- Updating `bin/web-server-text`
- Updating `bin/pdf-slideshow`
- Updating `bin/slideshow`

Setup-plt: Setup version is 202.5
Setup-plt: PLT home directory is `/home/mflatt/proj/plt`
Setup-plt: Collection Paths are: `/home/mflatt/proj/plt/collate`

There is a button labeled "Stop Installation".
More Process Examples
More Process Examples

gregs@ai.mit.edu (Gregory T. Sullivan)
To: L11 Discussion <ll1-discuss@ai.mit.edu>
Subject: L12, Sat. Nov. 9, 2002 - webcast info.
Date: 06 Nov 2002 11:51:57 -0500

Thanks to the generous sponsorship of Microsoft and Dr. Dobbs Journal, we will be able to webcast the L12 proceedings this coming Saturday, Nov. 9. The morning session (10am to 12:35pm EST) can be viewed at:

http://web.mit.edu/webcast/aailab/mit-l12-s1-09nov02-80k.ram

and the afternoon, from 1:45pm to 6pm will be at:

http://web.mit.edu/webcast/aailab/mit-l12-s2-09nov02-80k.ram

No new mail
Motivation and Approach

PLT Scheme as an Operating System

- Threads
- Parameters
- Eventspaces
- Custodians

Putting the Pieces to Work
Threads

Concurrent execution

(require "spin-display.scm")

(define (spin)
  (rotate-a-little)
  (sleep 0.1)
  (spin))

(define spinner (thread spin))

(kill-thread spinner)
Parameters (a.k.a. Fluid Variables)

Thread-local state

(printf "Hello\n")
(fprintf (current-output-port) "Hola\n")
(fprintf (current-error-port) "Goodbye\n")
(error "Ciao") eval

(parameterize ((current-error-port (current-output-port)))
  (error "Au Revoir")) eval

(parameterize ((current-error-port (current-output-port)))
  (thread
    (lambda ()
      (error "Zai Jian")))) eval
Eventspaces

Concurrent GUIs

(thread (lambda () (message-box "One" "Hi")))
(thread (lambda () (message-box "Two" "Bye")))

(thread (lambda () (message-box "One" "Hi")))
(parameterize ((current-eventspace (make-eventspace)))
  (thread (lambda () (message-box "Two" "Bye"))))
Custodians

Termination and clean-up

(define c (make-custodian))
(parameterize ((current-custodian c)) ...)  eval

(custodian-shutdown-all c)  eval
Custodians

Resource limits

(define (run-away)
  (cons 1 (run-away)))

(custodian-limit-memory c 1000000 c)

(parameterize ((current-custodian c))
  ...
  (thread run-away))
Etc.

- Security Guards
  - Resource access control
- Namespaces
  - Global bindings
- Will Executors
  - Timing of finalizations
- Inspectors
  - Debugging access
Motivation and Approach

PLT Scheme as an Operating System

Putting the Pieces to Work

- SchemeEsq, a mini DrScheme [ICFP99]
(define frame
  (new frame%
      [label "SchemeEsq"]
      [width 400] [height 175]))

(send frame show #t)
GUI - Reset Button

(new button
  [label "Reset"]
  [parent frame]
  [callback (lambda (b e) (reset-program))])

 eval
GUI - Interaction Area

(define repl-display-canvas
  (new editor-canvas% [parent frame]))

(eval)
(define esq-text% (class text% ... (evaluate str) ...))

(define repl-editor (new esq-text%))
(send repl-display-canvas set-editor repl-editor)

(eval)
Evaluator

(define (evaluate expr-str)
  (thread
   (lambda ()
     (print (eval (read (open-input-string expr-str))))
     (newline)
     (send repl-editor new-prompt))))
(define user-output-port
  (make-custom-output-port ... repl-editor ...))

(define (evaluate expr-str)
  (parameterize ((current-output-port user-output-port))
    (thread
      (lambda ()
        (...))))))
Evaluating GUls

(define user-eventspace (make-eventspace))

(define (evaluate expr-str)
  (parameterize ((current-output-port user-output-port)
                 (current-eventspace user-eventspace))
    (thread
     (lambda ()
       ...))))
(define user-custodian (make-custodian))

(define user-eventspace
  (parameterize ((current-custodian user-custodian))
    (make-eventspace)))

(define (evaluate expr-str)
  (parameterize ((current-output-port user-output-port)
                 (current-eventspace user-eventspace)
                 (current-custodian user-custodian))
    (thread
     (lambda ()
        ...))))

eval
(define (reset-program)
  (custodian-shutdown-all user-custodian)

  (set! user-custodian (make-custodian))
  (parameterize ((current-custodian user-custodian))
    (set! user-eventspace (make-eventspace)))
  (send repl-editor reset))
Conclusion

- Programmers need OS-like constructs in languages
  - concurrency
  - adjust run-time environment
  - easy termination

- Multiple language constructs for "process"
  - programmer can mix and match to balance isolation and cooperation