GNU Guile

Free Software Means of Production

FSCONS 2011

Andy Wingo

Greetings!

Andy Wingo

Guile co-maintainer, along with Ludovic Courtès

Agenda

Hacking Guile in 5 easy steps

On a mission: Guile & GNU

Live-hack!

Hacking Guile in 5 Easy Steps

Step one: Get Guile

Versions and Versions

2.0 is the awesomeness

1.8 is likely installed on your system

2.0 packages available for Fedora, Debian

Brief History

1995-1997: 1.3: An Emacs Lisp for the rest of GNU

1997-2002: 1.6: Adolescence

2002-2006: 1.8: Maturity

2007: Near-death: only 150 commits!

2008-2011: 2.0: Reactivation

Hacking Guile in 5 Easy Steps

Get Guile: Check!

Step two: Rock the REPL

REPL

```
define  loop
  print  eval  read
  loop
```

Guile's REPL has a lot more:

* Compiler and disassembler
* Profiler
* Tracer
* Debugger

Your program, alive

A Syntactic Interlude

```
(define (loop)
  (print (eval (read)))
  (loop))
```

Lisp: Lots of Irritating, Silly Parentheses?

Curly Braces?

```
var next = (function (){
  var x = 0;
  return function () {
    x = x + 1;
    return x;
  };
})();
```

;};})(); ?

Really?

Hello Parens, My Old Friends

```
(define next
  (let ((x 0))
    (lambda ()
      (set! x (+ x 1))
      x)))
```

‘Let’ and ‘define’ bind values to identifiers.

‘Lambda’ makes a function.

‘Set!’ sets a variable.

Bare identifiers return their bound values.

Anything else is a procedure call: ‘(+ x 1)’.

Hacking Guile in 5 Easy Steps

Get Guile: Check!

Rock the REPL: Check!

Step three: Use a proper editor

Proper Editors

Paren-matching

Indentation

Syntax highlighting

VIM and Emacs both qualify

A Stylistic Interlude

No dangling parens, please:

```
(define next
  (let ((x 0))
    (lambda ()
      (set! x (+ x 1))
      x
    )
  )
)
```

http://mumble.net/~campbell/scheme/style.txt

Hacking Guile in 5 Easy Steps

Get Guile: Check!

Rock the REPL: Check!

Use a proper editor: Check!

Structural editing

Paredit

Structural Editing for Emacs

[Demo]

Scheme's uniform structure facilitates higher-level editing operations

http://www.emacswiki.org/emacs/ParEdit

Hacking Guile in 5 Easy Steps

Get Guile: Check!

Rock the REPL: Check!

Use a proper editor: Check!

Structural editing: Check!

Live development

Geiser: Emacs Comes Alive

Extend running programs; incrementally build new programs

* Tab-completion
* Autodoc
* Live REPL, live eval (and redefinition)
* Who-calls, definition-at-point
* TCP to existing process or subprocess

http://www.nongnu.org/geiser/

Hacking Guile in 5 Easy Steps

Get Guile: Check!

Rock the REPL: Check!

Use a proper editor: Check!

Structural editing: Check!

Live development: Check!

Hacking Guile: Achievement unlocked!

Means of Production

Guile is a Scheme on a mission:

* Technical excellence in GNU
* GCC : Static :: Guile : Dynamic
* Well-suited to today's problems
* Fast

Technically Excellent

Delimited continuations! Building block for generators, coroutines, user-space preemptive threads

Rich data structures: Multidimensional typed numeric arrays, Unicode characters and strings, native data access

Macros: Embedded, compiled DSLs

Futures: Structured parallelism

First-class modules

A Collection of GNU Compilers

Guile: An HLVM

GCC for Scheme, Elisp, Lua

Specific facilities for dynamic languages

Redefinition of data, functions, classes (!)

Online compiler, debugger, reflective runtime

Language tower: Compile to Tree-IL, Guile takes care of the rest

Extending GNU

FFI (like Python's ctypes)

Good low-level POSIX bindings

Web modules: Server, client, URI, SXML

Native POSIX Threads (low-level and high-level abstractions)

Libraries (databases, GUI widgets, socket libs, etc)

Excellent C API

But Is It Fast?

Depends :)

Relative to CPython

Guile compiles to stack-machine bytecode

Bytecode interpreter (VM) written in C

Faster than default Python, Ruby implementations

Guile 2.2: Register VM, ~40% faster perhaps

Relative to GCC

Guile 2.0: About 40x slower than C

Register VM: 25x (perhaps)

Native code: 5x-10x (perhaps)

Achievable within 12 months

Further improvements require dynamic inlining, type feedback, aliasing analysis, vectorization

Let’s Hack!

@mattmight: Shorter *is* better. Let's skip to the logical conclusion--a service called "bitter" that allows only 1-bit tweets.

Strategy:

* Start in a guile --listen
* Experiment on the console
* Move to Emacs

Conclusion

Give Guile a try in your next project

Buy the fine manual! (Or just read it online)

Mailing list: guile-user@gnu.org

IRC: #guile on freenode

Bugs: bug-guile@gnu.org (no subscription req'd)

Thanks for listening

* http://gnu.org/s/guile/
* http://wingolog.org/