Rebuilding Guile from the ground up

A preview of Guile 2.2

GNU Hackers' Meeting 2014

Andy Wingo

wingo@igalia.com

Hacking compiler tech at Igalia since 2011

Mostly JS engine feature work (V8, SpiderMonkey)

Some work time on Guile

How we got here

Guile – GNU implementation of Scheme

Guile 2.0.0 in February 2011

Addition of compiler to Guile

- Stack VM
- Custom object file format
- Beginnings of optimized Scheme implementation

example.scm

```
(define (greet conf)
  (format #t "Hello, ~a!\n" conf))

(greet "GHM 2014")
```

Compilation can happen ahead of time, or as needed

```
$ guile-2.0 /tmp/example.scm
;;; note: auto-compilation is enabled, set GUILE_AUTO_COMPILE=0
;;;       or pass the --no-auto-compile argument to disable.
;;; compiling /tmp/example.scm
;;; compiled ~/.cache/guile/ccache/2.0-LE-8-2.0/tmp/example.scm.go
Hello, GHM 2014!
```

example.scm.go: alloc objtable

```
$ guild disassemble example.scm.go
Disassembly of #<objcode 7fe9985d5018>:

   0    (assert-nargs-ee/locals 8)      ;; 0 args, 1 local
   2    (make-false)
   3    (make-false)
   4    (make-false)
   5    (make-false)
   6    (make-false)
   7    (vector 0 5)                    ;; 5 elements
  10    (local-set 0)
  12    (new-frame)
  13    (local-ref 0)
  15    (load-program #{172}#)
  90    (local-ref 0)
  92    (call 1)
  95    (load-program #{173}#)
 357    (tail-call 0)
```

example.scm.go: fill objtable

```
Embedded program #{172}#:

   0    (assert-nargs-ee/locals 1)       ;; 1 arg, 0 locals
   2    (local-ref 0)
   4    (dup)
   5    (make-int8:1)                    ;; 1
   6    (load-symbol "format")           ;; format
  16    (vector-set)
  17    (dup)
  18    (make-int8 2)                    ;; 2
  20    (load-string "Hello, ~a!\n")     ;; "Hello, ~a!\n"
  35    (vector-set)
  36    (dup)
  37    (make-int8 3)                    ;; 3
  39    (load-symbol "greet")            ;; greet
  48    (vector-set)
  49    (dup)
  50    (make-int8 4)                    ;; 4
  52    (load-string "GHM 2014")         ;; "GHM 2014"
  64    (vector-set)
```

example.scm.go: “main”

```
Embedded program #{173}#:

   0    (assert-nargs-ee/locals 0)       ;; 0 args, 0 locals
   2    (make-false)
   3    (object-ref 1)
   5    (object-ref 2)
   7    (vector 0 3)                     ;; 3 elements
  15    (load-program #{174}#)
 156    (object-ref 3)
 158    (define)
 159    (object-ref 3)
 161    (link-now)
 162    (variable-ref)
 163    (object-ref 4)
 165    (tail-call 1)
```

example.scm.go: “greet”

```
Embedded program #{174}#:

   0    (assert-nargs-ee/locals 1)       ;; 1 arg, 0 locals
   2    (toplevel-ref 1)
   4    (make-true)                      ;; #t
   5    (object-ref 2)
   7    (local-ref 0)
   9    (tail-call 3)
```

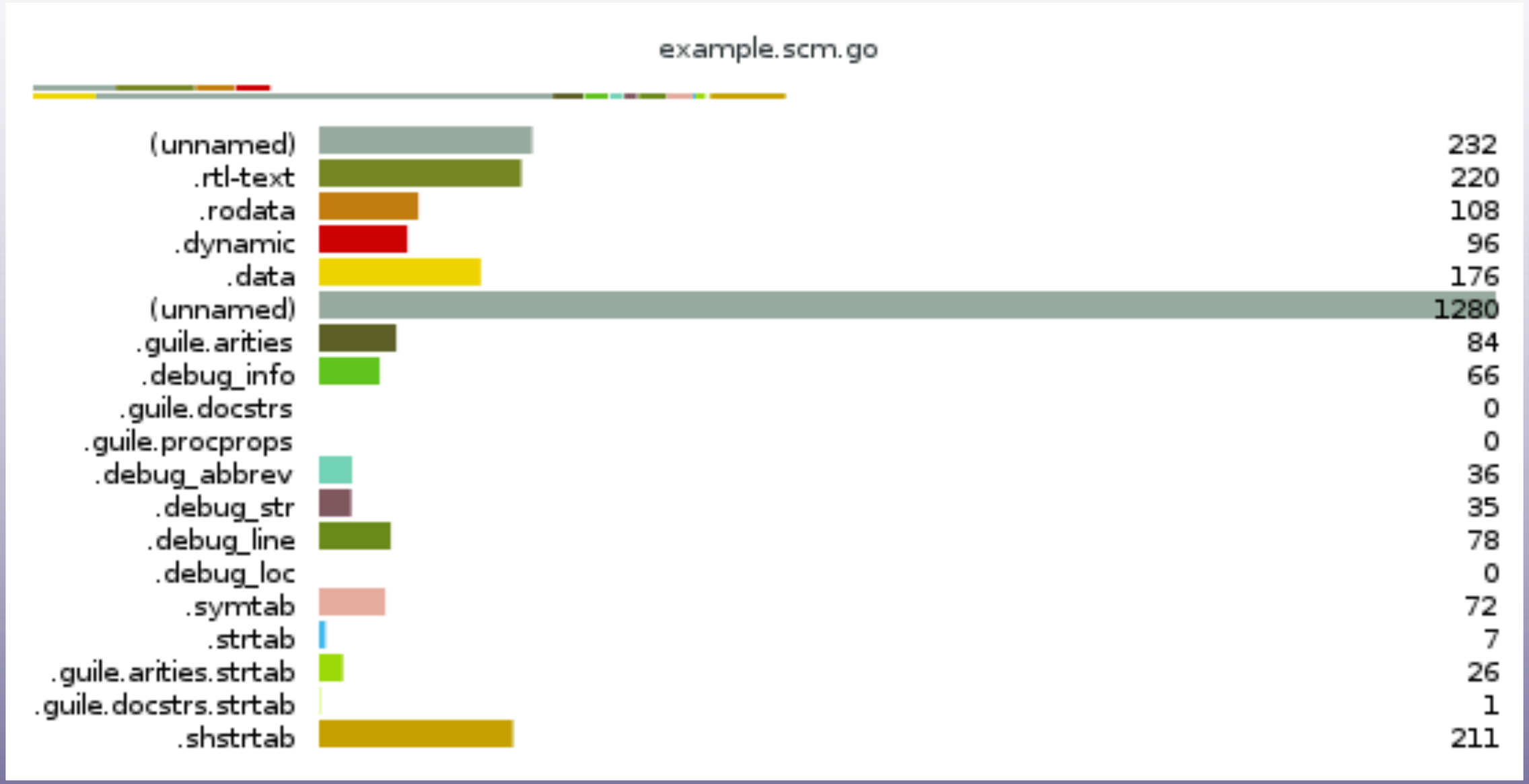
Needless work

Most constants don’t need to be on the heap – can be statically allocated in binary

Need binary format that can statically allocate constants

Shareable read-only data

ELF to the rescue



ELF to the rescue

Summary: linker allocates data statically

Thunk still run to relocate some links at run- -time, allocate needed heap data (symbols)

Needless work

Guile 2.0 stack VM means lots of instructions to shuffle operands and results into and out of local variables

Penalizes named local variables

Solution: register VM

`http://www.gnu.org/software/guile/docs/master/guile.html/A-Virtual-Machine-for-Guile.html`

[loop example]

Optimizing compiler

Contification

Type folding

CSE

Scalar replacement

More

(Match example)

Future

2.1.1 prerelease of 2.2 any day now

2.2 when it’s done

Q & A

Questions?

wingo@igalia.com wingo@pobox.com

http://wingolog.org/