Self-Hosted Scripting in Guile

Andy Wingo
Igalia, S.L.
Spain
wingo@igalia.com

ABSTRACT

Guile is a language implementation that grew out of the scripting movement in the 90s. Its initial implementation was entirely written in C. To increase the speed and expressive power of user programs written in Scheme, about 10 years ago Guile started to move to be a self-hosted compiler. Guile managed to keep the interactive, source-focussed, fast-starting aspects of scripting even while adding a sizeable Scheme compiler by making use of object file formats from the static compilation world: DWARF and ELF.

CCS CONCEPTS

• **Software and its engineering → Runtime environments**;

KEYWORDS

Self-hosting, startup time, scripting, object file formats

ACM Reference Format:
Andy Wingo. 2018. Self-Hosted Scripting in Guile. In *Proceedings of 2nd International Conference on the Art, Science, and Engineering of Programming (<Programming'18> Companion).* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3191697.3191727

1 PROBLEM STATEMENT

"Scripting" is an approach to programming that emphasizes a tight loop between the user, an external system such as the UNIX shell, and "scripts". These scripts are programs that are understood by the user as editable source code rather than untouchable build products.

Fast startup is essential to good user experience in scripting. Running a script should be perceived by the user to be essentially instantaneous: 10 or 20 milliseconds. This leaves not very much time to load the language runtime, not to mention reading, compiling, and running the script in question.

Guile [4] is an implementation of Scheme [5]. Scripting in Scheme poses a unique problem for startup in that the syntax is programmable: Scheme's *macros* are subprograms, potentially defined by the script in question, that process the script itself before it is run [6]. Compare to Lua, for example, which needs run no Lua code before running a script. Guile's first compiler, in version 2.0, introducd cached compilation artifacts for scripts, amortizing not only the read time for programs, but the macro expansion time too.

As one moves from a source-based system to a compiled system, there are a number of ancillary bits of metadata to carry forward that aren't usually necessary to a program's normal execution: source locations, procedure names, argument and local variable names, live variable maps, and the like. Users expect to see good backtraces, for example, and may want an interactive debugger. However recording this metadata in object files may be at odds with the fast-start behavior that users expect from scripts.

2 ELF AND DWARF IN GUILE

Guile's first compiler embedded script metadata in the object files in an ad-hoc fashion. The new compiler in version 2.2 takes this farther by using ELF [2] as the compiled object file format [3]. Guile writes DWARF [1] debugging sections that map code locations to source locations, in a way that can be stripped out from the binaries using standard `objdump` UNIX utility.

Guile embeds its own ELF and DWARF readers and writers and uses them on all platforms. The benefit of ELF and DWARF is that they encourage VM writers down a path which has low startup overhead. Read-only code and data pages are naturally shareable between processes, and writable data pages are lazily paged in by copy-on-write. Using ELF encourages an attention to the cost of relocation at startup, which results in more relative references between statically allocated code and data rather than absolute references that may be more convenient but require more relocation cost.

This talk will look into specific ways in which Guile used ELF and DWARF to result in single-digit millisecond boot times while keeping debuggability.

More generally, self-hosting runtimes can be part of a virtuous cycle, where a more powerful compiler leads to more optimized code and faster boot, or a vicious cycle where the growing weight of the runtime leads to longer and longer startup time. Keeping in mind the object file format of residual compiled code prevented Guile from straying into vice.

REFERENCES

[1] TIS Committee et al. 1995. DWARF debugging information format specification version 2.0. (May 1995).
[2] TIS Committee et al. 1995. Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification Version 1.2. *TIS Committee* (1995).
[3] The GNU Project. 2013. Guile Manual: Object File Format. (Dec. 2013). Retrieved January 26, 2018 from https://gnu.org/s/guile/docs/master/guile.html/Object-File-Format.html
[4] The GNU Project. 2018. Guile programming language. (Jan. 2018). Retrieved January 26, 2018 from https://gnu.org/s/guile
[5] Michael Sperber, R Kent Dybvig, Matthew Flatt, Anton Van Straaten, Robby Findler, and Jacob Matthews. 2009. Revised6 report on the algorithmic language Scheme. *Journal of Functional Programming* 19, S1 (2009), 1–301.
[6] Oscar Waddell and R Kent Dybvig. 1999. Extending the scope of syntactic abstraction. In *Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages.* ACM, 203–215.