Greenspun's tenth rule

Greenspun's tenth rule of programming is an aphorism in computer programming and especially programming language circles that states:[1][2]

> Any sufficiently complicated C or Fortran program contains an ad hoc, informally-specified, bug-ridden, slow implementation of half of Common Lisp.

Overview

The rule expresses the opinion that the argued flexibility and extensibility designed into the programming language Lisp includes all functionality that is theoretically needed to write any complex computer program, and that the features required to develop and manage such complexity in other programming languages are equivalent to some subset of the methods used in Lisp.

Other programming languages, while claiming to be simpler, require programmers to reinvent in a haphazard way a significant amount of needed functionality that is present in Lisp as a standard, time-proven, base.

It can also be interpreted as a satiric critique of systems that include complex, highly configurable sub-systems.[3] Rather than including a custom interpreter for some domain-specific language, Greenspun's rule suggests using a widely accepted, fully featured language like Lisp.

Paul Graham also highlights the satiric nature of the concept, albeit based on real issues:

> That sounds like a joke, but it happens so often to varying degrees in large programming projects that there is a name for the phenomenon, Greenspun's Tenth Rule.[4]

The rule was written sometime around 1993 by Philip Greenspun. Although it is known as his tenth rule, there are in fact no preceding rules, only the tenth. The reason for this according to Greenspun:

> Sorry, Han-Wen [5], but there aren't 9 preceding laws. I was just trying to give the rule a memorable name.[6]
>
> — Philip Greenspun, http://philip.greenspun.com/bboard/q-and-a-fetch-msg?msg_id=000tgU

Hacker Robert Morris later declared a corollary, which clarifies the set of "sufficiently complicated" programs to which the rule applies:

> …including Common Lisp.[7]

This corollary jokingly refers to the fact that many Common Lisp implementations (especially those available in the early 1990s) depend upon a low-level core of compiled C, which sidesteps the issue of bootstrapping but may itself be somewhat variable in quality, at least compared to a cleanly self-hosting Common Lisp.[8]

Software engineer Stewart Milberger then started a proof of Morris' corollary:

> I stopped porting Open Inventor to Common Lisp because I was implementing a buggy ad-hoc poorly specified implementation of namespaces in Common Lisp (Scheming Pony's first step in a proof of Morris' Corollary of Greeenspun's 10th) as Open Inventor had done before C++ namespaces were supported, like templates, and multi-methods (still not, I think). Someone also tell Stroustrup to look at Common Lisp macros, and stop the template madness.[9]

This proof jokingly refers to the fact that Common Lisp has a relatively primitive system for partitioning symbols called 'packages' and also that there might be a Greenspun's 11, vis-a-vis C++.

See also

- Inner-platform effect
- Software Peter principle
- Turing tarpit
- Zawinski's law of software envelopment

References

1. "Philip Greenspun's Research" (https://web.archive.org/web/20090124231039/http://philip.greenspun.com/research/). 1990–2017. Archived from the original (http://philip.greenspun.com/research/) on 2009-01-24. Retrieved 2019-10-24.
2. Graham, Paul (May 2002). "Revenge of the Nerds" (http://www.paulgraham.com/icad.html). Retrieved 2019-10-24.
3. Greenspun's Tenth Rule, does every large project include a Lisp interpreter? (http://programmers.stackexchange.com/a/137679)
4. Graham, Paul (2004). *Hackers & Painters: Big Ideas from the Computer Age* (https://archive.org/details/hackerspaintersb00grah/page/198). O'Reilly. p. 198 (https://archive.org/details/hackerspaintersb00grah/page/198). ISBN 978-0-596-00662-4. (also at Google books (https://www.google.com/books/edition/Hackers_Painters/shycAgAAQBAJ?hl=en&gbpv=1&bsq=%22Any%20sufficiently%20complicated%20C%20or%20Fortran%20program%20contains%22))
5. [1] (https://github.com/hanwen) Han-Wen's Github Profile
6. 10th rule of programming (http://philip.greenspun.com/bboard/q-and-a-fetch-msg?msg_id=000tgU)
7. Paul Graham quotes (http://paulgraham.com/quotes.html).
8. Rhodes, Christophe (2008-05-15). "SBCL: a Sanely-Bootstrappable Common Lisp" (https://www.doc.gold.ac.uk/~mas01cr/papers/s32008/sbcl.pdf) (PDF). *Lecture Notes in Computer Science* (Self-Sustaining Systems: First Workshop). Retrieved 2016-10-24.
9. "[libre-riscv-dev] Vulkanizing" (http://lists.libre-riscv.org/pipermail/libre-riscv-dev/2020-February/004284.html).