The Evolution of Lua

Roberto Ierusalimschy
Department of Computer Science,
PUC-Rio, Rio de Janeiro, Brazil
roberto@inf.puc-rio.br

Luiz Henrique de Figueiredo

Small size: Adding Lua to an application does not bloat it. The whole Lua distribution, including source code, documentation, and binaries for some platforms, has always fit comfortably on a floppy disk. The tarball for Lua 5.1, which contains source code, documentation, and examples, takes 208K compressed and 835K uncompressed. The source contains around 17,000 lines of C. Under Linux, the Lua interpreter built with all standard Lua libraries takes 143K. The corresponding numbers for most other scripting languages are more than an order of magnitude larger, partially because Lua is primarily meant to be embedded into Linux, all r(so)-348(its)-347[official distribution includes only a few libraries. Other scripting languages are meant to be used standalone and include many libraries.

Efficiency: Independent benchmarks [1] show Lua to be one the fastest realm interpreted scripting languages. This allows application developers to write a substantial fraction whole application in Lua. For instance, over 40% of Adobe Lightroom is

programming languages. The important difference — and what made DEL suitable for the data-entry problem — is that

	1.0	1.1	2.1	2.2	2.4	2.5	3.0	3.1	3.2	4.0	5.0	5.1
constructors	•	•	•	•	•	•	•	•	•	•	•	•
garbage collection	•	•	•	•	•	•	•	•	•	•	•	•
extensible semantics	◦	◦	•	•	•	•	•	•	•	•	•	•
support for OOP	◦	◦	•	•	•	•	•	•	•	•	•	•
long strings	◦	◦	◦	•	•	•	•	•	•	•	•	•
debug API	◦	◦	◦	•	•	•	•	•	•	•	•	•
external compiler	◦	◦	◦	◦	•	•	•	•	•	•	•	•
vararg functions	◦	◦	◦	◦	◦	•	•	•	•	•	•	•
pattern matching	◦	◦	◦	◦	◦	•	•	•	•	•	•	•
conditional compilation	◦	◦	◦	◦	◦	◦	•	•	•	◦	◦	◦

general *fallback*

academic circles.[6] In December 1996, shortly after Lua 2.5 was released, the magazine *Dr. Dobb's Journal* featured an article about Lua [16]. *Dr. Dobb's Journal* is a popular

5.4 Lua 4

Packaging library functions inside tables had a big practical impact, because it affected any program that used at least one library function. For instance, the old [illegible]

[illegible]

ever, such a change would probably break many existing

were moved to fields inside tables (see *§*

Practically all API functions get their operands from the

We could quite easily implement the original reference

However, Lua 3 behavior had a major drawback: it combined into a single primitive (`lua_pushuserdata`) two basic operations: userdata searching and userdata creation. For instance, it was impossible to check whether a given C pointer had a corresponding userdata without creating that userdata. Also, it was impossible to create a new userdata regardless of its C pointer. If Lua already had a userdata with that value, no new userdata would be created.

Lua 4 mitigated that drawback by introducing a new function, `lua_newuserdata`. Unlike `lua_pushuserdatax.n`,

istics of Lua. We have resisted user pressure to include other data structures, mainly “real” arrays and tuples, first by being stubborn, but also by providing tables with an efficient implementation and a flexible design. For instance, we can

use it! By far the most popular request was for a full macro

[29] R. Ierusalimschy, W. Celes, L. H. de Figueiredo, and