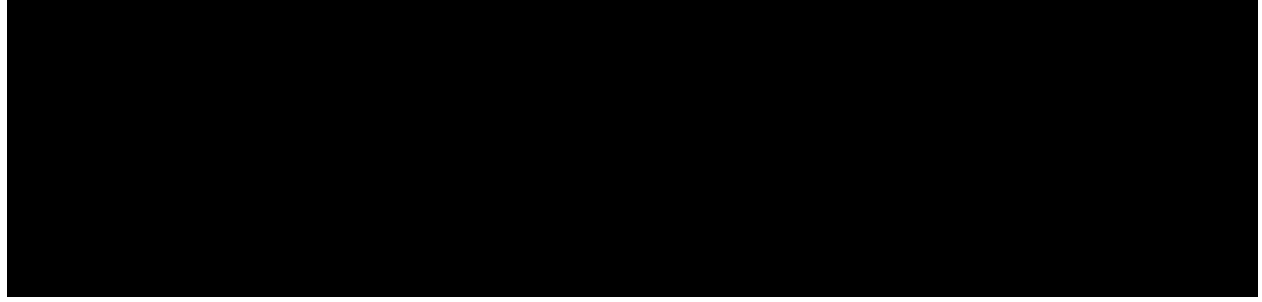


The Novelties of Lua 5.2

Roberto Ierusalimschy

Long list of changes

- a myriad of small improvements
- light C functions
- emergencGimpbagelightfuntions

Light C Functions

C functions without upvalues are stored as simple values, without memory allocation

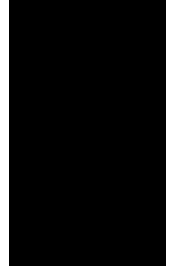
Light C Functions

- only possible due to change in environments
- new internal type

Emergency Garbage Collection

when memory allocation fails, collector does a complete collection cycle and then tries again

bitlib



Yieldable20.20.7RG0g0G0g0G0g0G0g0G1g1G0g0G0scall/me

Generational Collector

- basic idea: only young objects are traversed/collected
- *infant mortality* or *generational hypothesis*
  - good: less work when traversing objects
  - bad: less memory collected
- implementation uses the same apparatus of the incremental collector
  -

goto

- goto ts nicely with Lua philosophy of \mechanisms instead of policies"
  - very powerful mechanism
  - easy to explain
- allows the implementation of several mechanisms
  -

- break

`goto` implementation

- quite simple for the VM
  - small change to unify `OP_CLOSE` and `OP_JMP`
- parser must keep pending gotos and visible labels
- visibility rules
- closing of upvalues
- `break` implemented as `goto break`
  - each loop followes5tbilityandsmalllaband

Isn't `goto` evil?

-

_ENV

- the new scheme, with _ENV

Modules

- no more `module` function
- in general, less implicit things
- modules must explicitly change their environment and return their tables
- modules do not create globals by default
  - small problems with `-l` option for Lua stand-alone
  - common use: `local mod = require'mod'`

Macros in the large

- modularization
  - I

Conclusions

- a few long-wanted features
  - yieldable pcall/metamethods