Lua for the lazy C developer

Frank Vanbever <frank.vanbever@mind.be>
FOSDEM 2023
Who am I

- Frank Vanbever
- Embedded software developer for Mind (we’re hiring!)
  - FOSS for embedded systems
Why am I here?
Laziness: The quality that makes you go to great effort to reduce overall energy expenditure. It makes you write labor-saving programs that other people will find useful and document what you wrote so you don't have to answer so many questions about it.

https://thethreevirtues.com/
Lua is ...

- A programming language
- **Multi-paradigm**
  - Object-oriented (prototype inheritance)
  - Functional (first class functions)
- Dynamically typed
- Small (250K)
  - Small set of *meta-features* that allow you to implement what you need
  - 1 data structure: table
- Garbage collected
- a C library
Hello, World!

```lua
print 'hello world!'
lua_getglobal(L, "print");
lua_pushstring(L, "Hello, World");
lua_pcall(L, 1, 0, 0) != LUA_OK);
```
API - The Stack

- Lua is both an *extension* - (Lua from C) and an *extensible* (C from Lua) language
- Nearly all functions manipulate *The Stack*
- Fixes impedance mismatch
  - Static typed ↔ dynamic typed
  - Manual memory management ↔ garbage collection
Where Lua might make sense

● Taking care of tedious stuff that runs sporadically
  ○ “Look at me doing string manipulations in C, like an animal”
  ○ E.g. Config files in Wireplumber
● Prototyping
  ○ “I got some requirements communicated to me through interpretative dance”
  ○ REPL = superpower when doing discovery
● Plugins/Extensibility
  ○ “So what you want to do is you want to build a shared object against this specific libc…”
  ○ Get people to help themselves
  ○ E.g. swupdate handlers
function add(a, b)
    return a + b
end

lua_State *L = luaL_newstate();
lual_loadfile(L, "add.lua");
lua_pcall(L, 0, 0, 0);
lua_getglobal(L, "add");
lua_pushinteger(L, 1);
lua_pushinteger(L, 2);
lua_pcall(L, 2, 1, 0);
ret = lua_tointeger(L, -1);
local arithmetic = require("arithmetic")
local a = 3
local b = 3
local c = arithmetic.multiply(a, b)
print(c)

static int multiply(lua_State *L)
{
    int a = lua_tointeger(L, 1);
    int b = lua_tointeger(L, 2);
    lua_pushinteger(L, a*b);
    return 1;
}
static const struct luaL_Reg arithmetic[] = {
    {"multiply", multiply },
    {NULL, NULL}
};

int luaopen_arithmetic(lua_State *L)
{
    luaL_newlib(L, arithmetic);
    return 1;
}
function lua_subtract(a, b)
    return c_arithmetic.subtract(a, b)
end

static int subtract(lua_State *L) {
    int a = lua_tointeger(L, 1);
    int b = lua_tointeger(L, 2);
    lua_pushinteger(L, a-b);
    return 1;
}

static const struct luaL_Reg c_arithmetic[] = {
    {"subtract", subtract},
    {NULL, NULL},
};
lual_newlib(L, c_arithmetic);
lua_setglobal(L, "c_arithmetic");
// same as Lua from C
In short: Lua can help you get more done quicker embody the virtue of laziness
Some example code

https://gitlab.com/fvb/lua4lazyc