Binding With Members And Methods

Description

This is an example of how to make a Lua binding with *index* and *newindex* events to access members of a C structure.

The *newindex* event handler function has a lookup table in an upvalue with a list of all the members that can be set. For each member that can be set, the lookup table contains the offset of the member in the structure, and a pointer a glue function to set that type of member. We only have to write one glue function for each type of member: `double`, `int`, `char*`. This way we can use the same set `int` glue function for all the `int`s in this structure, or any other structure exposed to Lua.

The *index* event handler function is similar. It has a lookup table for all the members that you can get, but it also has a second lookup for all the method functions.

C code

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>

#include "lua.h"
#include "lauxlib.h"
#include "lualib.h"

/*
** {======================================================================
**   Example Lua bindings with members and methods
** =======================================================================
*/

static int get_int (lua_State *L, void *v)
{
  lua_pushnumber(L, *(int*)v);
  return 1;
}

static int set_int (lua_State *L, void *v)
```

```
{
  *(int*)v = luaL_checkint(L, 3);
  return 0;
}

static int get_number (lua_State *L, void *v)
{
  lua_pushnumber(L, *(lua_Number*)v);
  return 1;
}

static int set_number (lua_State *L, void *v)
{
  *(lua_Number*)v = luaL_checknumber(L, 3);
  return 0;
}

static int get_string (lua_State *L, void *v)
{
  lua_pushstring(L, (char*)v );
  return 1;
}

typedef int (*Xet_func) (lua_State *L, void *v);

/* member info for get and set handlers */
typedef const struct Xet_reg {
  const char *name;  /* member name */
  Xet_func func;     /* get or set function for type of member */
  size_t offset;     /* offset of member within your_t */
} *Xet_reg;

static void Xet_add (lua_State *L, Xet_reg l)
{
  for (; l->name; l++) {
    lua_pushstring(L, l->name);
    lua_pushlightuserdata(L, (void*)l);
    lua_settable(L, -3);
  }
}

static int Xet_call (lua_State *L)
{
  /* for get: stack has userdata, index, lightuserdata */
  /* for set: stack has userdata, index, value, lightuserdata */
```

```
  Xet_reg m = (Xet_reg)lua_touserdata(L, -1);  /* member info */
  lua_pop(L, 1);                               /* drop lightuserdata */
  luaL_checktype(L, 1, LUA_TUSERDATA);
  return m->func(L, lua_touserdata(L, 1) + m->offset);
}

static int index_handler (lua_State *L)
{
  /* stack has userdata, index */
  lua_pushvalue(L, 2);                     /* dup index */
  lua_rawget(L, lua_upvalueindex(1));      /* lookup member by name */
  if (!lua_islightuserdata(L, -1)) {
    lua_pop(L, 1);                         /* drop value */
    lua_pushvalue(L, 2);                   /* dup index */
    lua_gettable(L, lua_upvalueindex(2));  /* else try methods */
    if (lua_isnil(L, -1))                  /* invalid member */
      luaL_error(L, "cannot get member '%s'", lua_tostring(L, 2));
    return 1;
  }
  return Xet_call(L);                      /* call get function */
}

static int newindex_handler (lua_State *L)
{
  /* stack has userdata, index, value */
  lua_pushvalue(L, 2);                     /* dup index */
  lua_rawget(L, lua_upvalueindex(1));      /* lookup member by name */
  if (!lua_islightuserdata(L, -1))         /* invalid member */
    luaL_error(L, "cannot set member '%s'", lua_tostring(L, 2));
  return Xet_call(L);                      /* call set function */
}

/*
** ============================================================================
**  Your structure and custom methods
** ============================================================================
*/

#define YOUR_T "YourClass"

typedef struct {
  int    id;
  char   name[16];
  int    age;
  double x,y;
```

```
} your_t;


static your_t *check_your_t (lua_State *L, int index)
{
  your_t *yd;
  luaL_checktype(L, index, LUA_TUSERDATA);
  yd = (your_t *)luaL_checkudata(L, index, YOUR_T);
  if (yd == NULL) luaL_typerror(L, index, YOUR_T);
  return yd;
}

static your_t *push_your_t (lua_State *L)
{
  your_t *yd = (your_t*)lua_newuserdata(L, sizeof(your_t));
  luaL_getmetatable(L, YOUR_T);
  lua_setmetatable(L, -2);
  return yd;
}

static int id_counter;

static int your_create (lua_State *L)
{
  your_t *yd;
  int name_len;
  const char *name = luaL_checklstring(L, 1, &name_len);
  if( name_len > 15 ) luaL_error(L, "name too long"); /* die */
  yd = push_your_t(L);
  strcpy( yd->name, name );
  yd->age = luaL_checkint(L, 2);
  yd->x   = luaL_checknumber(L, 3);
  yd->y   = luaL_checknumber(L, 4);
  yd->id  = ++id_counter;
  return 1;
}

static int your_destroy (lua_State *L)
{
  your_t *yd = (your_t*)lua_touserdata(L,1);
  printf("Goodbye %s:%d at (%lf,%lf)\n", yd->name, yd->id, yd->x, yd->y);
  return 0;
}

static int your_position (lua_State *L)
```

```
{
  your_t *yd = check_your_t(L, 1);
  double   x = yd->x;
  double   y = yd->y;
  if( lua_gettop(L) > 1 ) {
    yd->x = luaL_checknumber(L, 2);
    yd->y = luaL_checknumber(L, 3);
  }
  lua_pushnumber(L,x);
  lua_pushnumber(L,y);
  return 2;
}

static int test (lua_State *L)
{
  int n = luaL_checknumber(L, 1);
  lua_pushnumber(L, 66);
  lua_pushnumber(L, 67);
  lua_pushnumber(L, 68);
  return n;
}


static const luaL_reg your_meta_methods[] = {
{"__gc", your_destroy },
{0,0}
};

static const luaL_reg your_methods[] = {
{"create",   your_create},
{"position", your_position},
{"test",     test},
{0,0}
};

static const struct Xet_reg your_getters[] = {
{"id",   get_int,    offsetof(your_t,id)   },
{"name", get_string, offsetof(your_t,name) },
{"age",  get_int,    offsetof(your_t,age)  },
{"x",    get_number, offsetof(your_t,x)    },
{"y",    get_number, offsetof(your_t,y)    },
{0,0}
};

static const struct Xet_reg your_setters[] = {
```

```
{"age",  set_int,    offsetof(your_t,age)  },
{"x",    set_number, offsetof(your_t,x)    },
{"y",    set_number, offsetof(your_t,y)    },
{0,0}
};


int your_register (lua_State *L)
{
  int metatable, methods;

  /* create methods table, & add it to the table of globals */
  luaL_openlib(L, YOUR_T, your_methods, 0);
  methods = lua_gettop(L);

  /* create metatable for your_t, & add it to the registry */
  luaL_newmetatable(L, YOUR_T);
  luaL_openlib(L, 0, your_meta_methods, 0);  /* fill metatable */
  metatable = lua_gettop(L);

  lua_pushliteral(L, "__metatable");
  lua_pushvalue(L, methods);    /* dup methods table*/
  lua_rawset(L, metatable);     /* hide metatable:
                                   metatable.__metatable = methods */
  lua_pushliteral(L, "__index");
  lua_pushvalue(L, metatable);  /* upvalue index 1 */
  Xet_add(L, your_getters);     /* fill metatable with getters */
  lua_pushvalue(L, methods);    /* upvalue index 2 */
  lua_pushcclosure(L, index_handler, 2);
  lua_rawset(L, metatable);     /* metatable.__index = index_handler */

  lua_pushliteral(L, "__newindex");
  lua_newtable(L);              /* table for members you can set */
  Xet_add(L, your_setters);     /* fill with setters */
  lua_pushcclosure(L, newindex_handler, 1);
  lua_rawset(L, metatable);     /* metatable.__newindex = newindex_handler */

  lua_pop(L, 1);                /* drop metatable */
  return 1;                     /* return methods on the stack */
}


int main(int argc, char *argv[])
{
  lua_State *L = lua_open();
```

```
  luaopen_base(L);
  luaopen_table(L);
  luaopen_io(L);
  luaopen_string(L);
  luaopen_math(L);
  luaopen_debug(L);

  your_register(L);

  if(argc>1) lua_dofile(L, argv[1]);

  lua_close(L);
  return 0;
}
```

Compiling the Code

This code can be compiled for Lua 5.0 as follows:

```
gcc member.c  -L/usr/local/lib/ -llua -llualib
```

Lua Test Code

```
print('= YourClass =',YourClass)
for n,v in YourClass do print(n,v) end

b = YourClass.create('bill', 99, 34,65)
f = YourClass.create('fish', 44, 16.7,19.25)

local member_names = { 'id','name','age','x','y' }

function dump( ud )
  for _,n in member_names do print(n, ud[n]) end
end

print'*** b ***'
dump(b)

print'*** f ***'
dump(f)

print('b.name =', b.name, 'pos = ', b.x, b.y)
```

```
b.age = 88
b.x = 40
b.y = 50

print'*** modified b ***'
dump(b)

debug.debug()
```

Test Code Output

```
$ ./a member.lua
= YourClass =   table: 0xa044f00
test    function: 0xa044f60
create  function: 0xa044f28
position        function: 0xa045240
*** b ***
id      1
name    bill
age     99
x       34
y       65
*** f ***
id      2
name    fish
age     44
x       16.7
y       19.25
b.name =        bill    pos =   34      65
*** modified b ***
id      1
name    bill
age     88
x       40
y       50
lua_debug> print(b.age, b:position())
88      40      50
lua_debug> b.age = 99
lua_debug> b.name = 'rat'
[string "b.name = 'rat'..."]:1: cannot set member 'name'
lua_debug> print(b.cow)
[string "print(b.cow)..."]:1: cannot get member 'cow'
lua_debug> b.fish = 9
[string "b.fish = 9..."]:1: cannot set member 'fish'
```

```
lua_debug> dump(b)
id      1
name    bill
age     99
x       40
y       50
lua_debug> cont
Goodbye fish:2 at (16.700000,19.250000)
Goodbye bill:1 at (40.000000,50.000000)
```

FindPage · RecentChanges · preferences
edit · history
Last edited May 15, 2003 5:33 pm PDT (diff)