User Data Example

Description

This Lua 5.0 example shows how to store a structure in a userdata.

Another appoach to store only a single pointer in the userdata. See UserDataWithPointerExample.

The metatable for the userdata is put in the registry, and the `__index` field points to the table of methods so that the `object:method()` syntax will work. The methods table is stored in the table of globals so that scripts can add methods written in Lua.

The Lua functions that use the stucture will need to either access a userdata on the stack, or push a new userdata onto the stack.

`checkFoo` ensures that a userdata on the stack is the correct type, and returns a pointer to the structure inside the userdata.

`pushFoo` leaves a new userdata on top of the stack, sets its metatable, and returns a pointer to the structure so that you can fill in the fields.

foo.c C code

```
#include "lua.h"
#include "lauxlib.h"

#define FOO "Foo"

typedef struct Foo {
  int x;
  int y;
} Foo;


static Foo *toFoo (lua_State *L, int index)
{
  Foo *bar = (Foo *)lua_touserdata(L, index);
  if (bar == NULL) luaL_typerror(L, index, FOO);
  return bar;
}
```

```
static Foo *checkFoo (lua_State *L, int index)
{
  Foo *bar;
  luaL_checktype(L, index, LUA_TUSERDATA);
  bar = (Foo *)luaL_checkudata(L, index, FOO);
  if (bar == NULL) luaL_typerror(L, index, FOO);
  return bar;
}

static Foo *pushFoo (lua_State *L)
{
  Foo *bar = (Foo *)lua_newuserdata(L, sizeof(Foo));
  luaL_getmetatable(L, FOO);
  lua_setmetatable(L, -2);
  return bar;
}


static int Foo_new (lua_State *L)
{
  int x = luaL_optint(L, 1, 0);
  int y = luaL_optint(L, 2, 0);
  Foo *bar = pushFoo(L);
  bar->x = x;
  bar->y = y;
  return 1;
}

static int Foo_yourCfunction (lua_State *L)
{
  Foo *bar = checkFoo(L, 1);
  printf("this is yourCfunction\t");
  lua_pushnumber(L, bar->x);
  lua_pushnumber(L, bar->y);
  return 2;
}

static int Foo_setx (lua_State *L)
{
  Foo *bar = checkFoo(L, 1);
  bar->x = luaL_checkint(L, 2);
  lua_settop(L, 1);
  return 1;
}
```

```
static int Foo_sety (lua_State *L)
{
  Foo *bar = checkFoo(L, 1);
  bar->y = luaL_checkint(L, 2);
  lua_settop(L, 1);
  return 1;
}

static int Foo_add (lua_State *L)
{
  Foo *bar1 = checkFoo(L, 1);
  Foo *bar2 = checkFoo(L, 2);
  Foo *sum  = pushFoo(L);
  sum->x = bar1->x + bar2->x;
  sum->y = bar1->y + bar2->y;
  return 1;
}

static int Foo_dot (lua_State *L)
{
  Foo *bar1 = checkFoo(L, 1);
  Foo *bar2 = checkFoo(L, 2);
  lua_pushnumber(L, bar1->x * bar2->x + bar1->y * bar2->y);
  return 1;
}

static const luaL_reg Foo_methods[] = {
  {"new",           Foo_new},
  {"yourCfunction", Foo_yourCfunction},
  {"setx",          Foo_setx},
  {"sety",          Foo_sety},
  {"add",           Foo_add},
  {"dot",           Foo_dot},
  {0, 0}
};

static int Foo_gc (lua_State *L)
{
  printf("bye, bye, bar = %p\n", toFoo(L, 1));
  return 0;
}

static int Foo_tostring (lua_State *L)
{
```

```
  char buff[32];
  sprintf(buff, "%p", toFoo(L, 1));
  lua_pushfstring(L, "Foo (%s)", buff);
  return 1;
}

static const luaL_reg Foo_meta[] = {
  {"__gc",       Foo_gc},
  {"__tostring", Foo_tostring},
  {"__add",      Foo_add},
  {0, 0}
};

int Foo_register (lua_State *L)
{
  luaL_openlib(L, FOO, Foo_methods, 0);  /* create methods table,
                                            add it to the globals */
  luaL_newmetatable(L, FOO);          /* create metatable for Foo,
                                         and add it to the Lua registry */
  luaL_openlib(L, 0, Foo_meta, 0);    /* fill metatable */
  lua_pushliteral(L, "__index");
  lua_pushvalue(L, -3);               /* dup methods table*/
  lua_rawset(L, -3);                  /* metatable.__index = methods */
  lua_pushliteral(L, "__metatable");
  lua_pushvalue(L, -3);               /* dup methods table*/
  lua_rawset(L, -3);                  /* hide metatable:
                                         metatable.__metatable = methods */
  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);

  Foo_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 foo.c  -L/usr/local/lib -llua -llualib
```

Lua Test Code

```
for n,v in Foo do print(n,v) end

local a = Foo.new()
local b = Foo.new(99,100)

MyFunction = Foo.yourCfunction

print( a, MyFunction(a) )
print( b, MyFunction(b) )


function Foo:show(msg)
  print( msg, self, self:yourCfunction() )
  return self
end

function Foo:point(t)
  assert(type(t) == 'table')
  self:setx(t.x or t[1]):sety(t.y or t[2])
  return self
end

setmetatable(Foo, { __call = function(self, x, y)
                               local bar = self.new(x,y)
                               print('created', bar)
                               return bar
                             end } )

local p = Foo(1,2)
```

```
p:show('p is')
p:setx(3):show'p is':sety(4):show'p is'
p:point{33,44}:show'p is'
p = nil

collectgarbage()

a:point{ x=500, y=1000}
a:show'a is'

r = Foo.add(a,b)
r:show'r is'

a:show'a is'
b:show'b is'
s = a + b
s:show's is'

--debug.debug()
```

Test Code Output

```
$ ./a test.lua
sety    function: 0xa045388
dot     function: 0xa045328
setx    function: 0xa044fb8
yourCfunction   function: 0xa0452c8
add     function: 0xa0452f8
new     function: 0xa044f80
this is yourCfunction   Foo (0xa046938) 0       0
this is yourCfunction   Foo (0xa046760) 99      100
created Foo (0xa045458)
this is yourCfunction   p is    Foo (0xa045458) 1       2
this is yourCfunction   p is    Foo (0xa045458) 3       2
this is yourCfunction   p is    Foo (0xa045458) 3       4
this is yourCfunction   p is    Foo (0xa045458) 33      44
bye, bye, bar = 0xa045458
this is yourCfunction   a is    Foo (0xa046938) 500     1000
this is yourCfunction   r is    Foo (0xa045478) 599     1100
this is yourCfunction   a is    Foo (0xa046938) 500     1000
this is yourCfunction   b is    Foo (0xa046760) 99      100
this is yourCfunction   s is    Foo (0xa046470) 599     1100
bye, bye, bar = 0xa046470
bye, bye, bar = 0xa045478
```

```
bye, bye, bar = 0xa046760
bye, bye, bar = 0xa046938
```

FindPage · RecentChanges · preferences
edit · history
Last edited July 31, 2003 10:27 am PDT (diff)