build failing

OpenSoundControl for Lua (https://web.archive.org/web/20160809090717/https://travis-ci.org/lubyk/osc)

OpenSoundControl pack and unpack for Lua.

(https://web.archive.org/web/20160809090717/https://github.com/lubyk/osc)

MIT license © Ross Bencina 2013, Gaspard Bucher 2014.

Web page for oscpack (https://web.archive.org/web/20160809090717/http://www.rossbencina.com/code/oscpack).

Installation

With luarocks (https://web.archive.org/web/20160809090717/http://luarocks.org/):

```
$ luarocks install osc
```

Supports sending basic Lua values and (nested) lua tables either as Array or Hash. A table with both numeric and string keys is treated as an array.

WARN This implementation does not support sending binary data.

Usage example

```
local osc = require 'osc'
local data = osc.pack('/some/url', true, 2, {foo = 'bar'})
-- ... send ... receive
local url, a, b, c = osc.unpack(data)
```

.VERSION `= '1.0.1'`

Current version respecting semantic versioning (https://web.archive.org/web/20160809090717/http://semver.org/).

.DEPENDS = {

`"lua >= 5.1, < 5.4"`

Compatible with Lua 5.1 to 5.3 and LuaJIT

`'lub >= 1.0.3, < 2.0'`

Uses Lubyk base library (https://web.archive.org/web/20160809090717/http://doc.lubyk.org/lub.html)

}

Class methods

.pack `(url, ...)`

Pack an url with values into a binary string ready to be transmitted.

```lua
local data = osc.pack(url, value1, value2)
```

.unpack `(data)`

Unpack binary data into lua values. This is a multi value return function:

```lua
local url, value1, value2 = osc.unpack(data)
```

Client (osc.Client.html)

This is a simple UDP client (based on lens.Socket) to send OSC messages.

Server (osc.Server.html)

This is a simple UDP server (based on lens.Socket) to receive OSC messages.

osc (index.html)

Client (osc.Client.html)

Server (osc.Server.html)

(https://web.archive.org/web/20160811014512/http://lubyk.org/)

Lubyk (https://web.archive.org/web/20160811014512/http://lubyk.org/) documentation

osc.Client

Class functions

Methods

osc send client

This is a simple UDP client (based on lens.Socket) to send OSC messages.

NOTE This class needs the lens (https://web.archive.org/web/20160811014512/http://doc.lubyk.org/lens.html) library.

Usage example

```
local osc = require 'osc'

client = osc.Client (osc.Client.html)('127.0.0.1', 11000)

client:send('/hello', 'lubyk', 2014)
```

Class functions

.new `(host, port)`

Create a new client connected to a given `host` and `port`.

Methods

:send `(url, ...)`

Send osc message.

osc (osc.html)

Client (osc.Client.html)

Server (osc.Server.html)

(https://web.archive.org/web/20160809090712/http://lubyk.org/)

Lubyk

(https://web.archive.org/web/20160809090712/http://lubyk.org/) documentation

osc.Server

Callback

---

osc receive server

This is a simple UDP server (based on lens.Socket) to receive OSC messages.

The server must be created and run inside lens.Scheduler (lens.Scheduler.html) (see example below).

NOTE This class needs the lens (https://web.archive.org/web/20160809090712/http://doc.lubyk.org/lens.html) library.

Usage example

```
local lens = require 'lens'
-- Using live coding
lens.run(function() lens.FileWatch (lens.FileWatch.html)() end)

local osc = require 'osc'

server = server or osc.Server (osc.Server.html)(11000)

function server:receive(url, ...)
  print(url, ...)
end
```

.new `(port, map)`

Create a new server. If `port` is '0', a random available port will be chosen. If an optional `map` table is provided, it is used to trigger functions from message url (see map).

:map `(map)`

Trigger functions from message urls. Calling this function overwrites the receive callback.

Example:

```
server:map {
  ['/1/fader1'] = function(url, value)
    print('HEY, fader 1 changed', value)
  end,

  ['/1/pad1'] = function(url, x, y)
    box:move(x, y)
  end,

  unknown = function(url, ...)
    print('Missing entry in map table', url, ...)
  end,
}
```

The 'unknown' entry is used to map all urls not present in the table.

Callback

:receive `(url, ...)`

This callback is called when osc messages arrive. If map is used, this callback is changed.

osc (osc.html)

Client (osc.Client.html)

Server (osc.Server.html)