Lua binding to libcurl

Status

This module include three layer

1. `lcurl` module provide low level pure C binding to libcurl.
   Almost ready and needs tests. I have no plans to change this API.
2. `cURL` module provide compatibility for Lua-cURLv2 API.
   Almost ready and needs tests.
3. `cURL` module provide new high level API.
   In fact for now it provide `lcurl` API directly and needed to redesign.

Documentation

lcurl API
Lua-cURLv2 API
Lua-cURLv3 API

Original Lua-cURLv2 binding has several problems:

- it can not return error codes but just raise Lua errors (Fixed. use `cURL.safe` module)
- it raise Lua error from callback that may result resource leak in libcurl (Fixed.)
- it does not provide building multipart/formdata explicitly (Fixed.)
- it has memory leak when send multipart/formdata (Fixed.)
- it does not save string for curl options that may result crush in libcurl (Fixed.)
- there no way to get result for operations in multi interface (e.g. if one of easy operation fail you can not get result code/error message) (Fixed. But it does not very handy interface.)
- you can not use your own callback function to perform operation with multi interface (Could not be fixed without changing API.)
- you can not pass your context to callback functions (Could not be fixed without changing API.)

Installation

Using LuaRocks:

```
luarocks install Lua-cURL
```

Install current master:

```
luarocks install Lua-cURL --server=https://luarocks.org/dev
```

List of incompatibility with original [Lua-cURLv2]

- objects are tables
- multi:perform() also returns ("done",code), ("error",error) and ("response",code) records
- writer callback does not recv string len (just string itself)
- on Lua > 5.2 errors are objects but not strings

Usage

```
-- HTTP Get
curl.easy{
    url = 'http://httpbin.org/get',
    httpheader = {
      "X-Test-Header1: Header-Data1",
      "X-Test-Header2: Header-Data2",
    },
    writefunction = io.stderr -- use io.stderr:write()
  }
  :perform()
:close()
```

```
-- HTTP Post
curl.easy()
  :setopt_url('http://posttestserver.com/post.php')
  :setopt_writefunction(io.write)
  :setopt_httppost(curl.form() -- Lua-cURL guarantee that form will be alive
    :add_content("test_content", "some data", {
      "MyHeader: SomeValue"
    })
    :add_buffer("test_file", "filename", "text data", "text/plain", {
      "Description: my file description"
    })
    :add_file("test_file2", "BuildLog.htm", "application/octet-stream", {
      "Description: my file description"
    })
  )
  :perform()
:close()
```

```
-- FTP Upload
local function get_bin_by(str,n)
  local pos = 1 - n
  return function()
    pos = pos + n
    return (str:sub(pos,pos+n-1))
  end
end

curl.easy()
  :setopt_url("ftp://moteus:123456@127.0.0.1/test.dat")
```

```
  :setopt_upload(true)
  :setopt_readfunction(
    get_bin_by(("0123456789"):rep(4), 9)
  )
  :perform()
:close()
```

```
-- Multi FTP Upload

-- We get error E_LOGIN_DENIED for this operation
e1 = curl.easy{url = "ftp://moteus:999999@127.0.0.1/test1.dat", upload = true}
  :setopt_readfunction(
    function(t) return table.remove(t) end, {"1111", "2222"}
  )

e2 = curl.easy{url = "ftp://moteus:123456@127.0.0.1/test2.dat", upload = true}
  :setopt_readfunction(get_bin_by(("e"):rep(1000), 5))

m = curl.multi()
m:add_handle(e1)
m:add_handle(e2)

while m:perform() > 0 do m:wait() end

while true do
  h, ok, err = m:info_read()
  if h == 0 then break end

  if h == e1 then
    assert(ok == nil)
    assert(err:name() == "LOGIN_DENIED")
    assert(err:no() == curl.E_LOGIN_DENIED)
  end

  if h == e2 then
    assert(ok == true)
  end
end
```