

The lualibs package

Élie Roux · elie.roux@telecom-bretagne.eu
Philipp Gesang · phg@phi-gamma.net
Ulrike Fischer · fischer@troubleshooting-tex.de

2019/02/14 v2.64

Abstract

Additional Lua functions taken from the `l-*` and `util-*` files of Con \TeX T. For an introduction on this package (among others), please refer to the document `lualatex-doc.pdf`.

Contents

I	Package Description	1
1	Overview	1
2	Usage	2
2.1	Loading Library Collections	2
2.2	Loading libraries Individually	2
3	Files	3
4	Packaging	3
II	<code>lualibs.lua</code>	4
III	<code>lualibs-basic.lua</code>	7
IV	<code>lualibs-extended.lua</code>	8

I Package Description

1 OVERVIEW

Lua is a very minimal language, and it does only have a minimal standard library. The aim of this package is to provide an extended standard library, to be used by various Lua \TeX packages. The code is specific to Lua \TeX and depends on Lua \TeX functions and modules not available in regular lua.

The code is derived from Con \TeX T libraries.

2 USAGE

You can either load the `lualibs` module, which will in turn load one of two sets of libraries provided by this package: `require("lualibs")`, or explicitly load the modules you need, e.g.: `require("lualibs-table")`, please note that some modules depend on others.

2.1 *Loading Library Collections*

The libraries are split into a basic and an extended collection. Though neither initialization time nor memory consumption will be noticeably impacted,¹ the `lualibs` package can skip loading of the latter on demand. The `config` table needs to be present prior to loading the package for this to work (in the future this may be achieved by an option of `\usepackage` for \LaTeX users). In the `lualibs` field, set `load_extended` to false:

```
\directlua{
  --- 1) create config table safely
  config          = config or { }
  config.lualibs  = config.lualibs or { }

  --- 2) unset the load_extended entry
  config.lualibs.load_extended = false

  --- 3) load the lualibs
  require "lualibs"
}
```

However, there is no guarantee that the extended set remains permanently excluded. Re-loading the package at a later point will cancel this option and possibly add the missing libraries.

2.2 *Loading libraries Individually*

In order to load the `table` module you would actually have to prepare it as follows:

```
require"lualibs-lua"
require"lualibs-lpeg"
require"lualibs-table"
```

If your code is run by the `texlua` interpreter, you will need to initialize `kpse` library so that `require()` can find files under `TEXMF` tree: `kpse.set_program_name("luatex")`.

¹Note that in terms of code this is only a small fraction of what `ConTeXt` loads with every run.

Table 1: The basic set.

lua _{libs} name	ConT _E Xt name	primary purpose
lua _{libs} -lua.lua	l-lua.lua	compatibility
lua _{libs} -package.lua	l-package.lua	Lua file loaders
lua _{libs} -lpeg.lua	l-lpeg.lua	patterns
lua _{libs} -function.lua	l-function.lua	defines a dummy function
lua _{libs} -string.lua	l-string.lua	string manipulation
lua _{libs} -table.lua	l-table.lua	serialization, conversion
lua _{libs} -boolean.lua	l-boolean.lua	boolean converter
lua _{libs} -number.lua	l-number.lua	bit operations
lua _{libs} -math.lua	l-math.lua	math functions
lua _{libs} -io.lua	l-io.lua	reading and writing files
lua _{libs} -os.lua	l-os.lua	platform specific code
lua _{libs} -file.lua	l-file.lua	filesystem operations
lua _{libs} -gzip.lua	l-gzip.lua	wrapper for lgzip
lua _{libs} -md5.lua	l-md5.lua	checksum functions
lua _{libs} -dir.lua	l-dir.lua	directory handling
lua _{libs} -unicode.lua	l-unicode.lua	utf and unicode
lua _{libs} -url.lua	l-url.lua	url handling
lua _{libs} -set.lua	l-set.lua	sets

3 FILES

The lua_{libs} bundle contains files from two ConT_EXt Lua library categories: The generic auxiliary functions (original file prefix: l-) together form something close to a standard library. Most of these are extensions of an existing namespace, like for instance l-table.lua which adds full-fledged serialization capabilities to the Lua table library. They were imported under the lua_{libs}-prefix and are contained in the basic collection. (For a list see table 1.)

The extended category comprises a selection of files mostly from the utilities namespace (util-; cf. table 2). Their purpose is more specific and at times quite low-level. Additionally, the file trac-inf.lua has been included because it is essential to some of the code loaded subsequently.

4 PACKAGING

By default, lua_{libs} will not load the libraries individually. Instead, it includes two *merged packages* that have been compiled from the original files. This is achieved by means of mt_x-package, a script for bundling Lua code shipped with ConT_EXt. This concatenates the code of several Lua files into a single file that is both easier to distribute and loading marginally faster. mt_x-package ensures that the code from each file gets its own closure and strips newlines and comments, resulting in a smaller payload. Another package that relies on it heavily is the font loader as contained in luaotfload and luatex-fonts. Luaot-

Table 2: The extended set.

lua _{libs} name	ConT _E Xt name	primary purpose
lua _{libs} -util-str.lua	util-str.lua	extra string functions
lua _{libs} -util-fil.lua	util-fil.lua	extra file functions
lua _{libs} -util-tab.lua	util-tab.lua	extra table functions
lua _{libs} -util-sto.lua	util-sto.lua	table allocation
lua _{libs} -util-prs.lua	util-sto.lua	miscellaneous parsers
lua _{libs} -util-dim.lua	util-dim.lua	conversion between dimensions
lua _{libs} -trac-inf.lua	trac-inf.lua	timing, statistics
lua _{libs} -util-lua.lua	util-lua.lua	operations on bytecode
lua _{libs} -util-deb.lua	util-deb.lua	extra debug functionality
lua _{libs} -util-tpl.lua	util-tpl.lua	templating
lua _{libs} -util-sta.lua	util-sta.lua	stacker (e. g. for PDF)
lua _{libs} -util-jsn.lua	util-jsn.lua	conversion to and from json

fload, a port of the ConT_EXt fontloader for Plain and L^AT_EX, has a hard dependency on the functionality provided by the Lua_{libs} package. The packages should not be updated independently.

If ConT_EXt is installed on the system, the merge files can be created by running:

```
mtxrun --script package --merge lualibs-basic.lua
mtxrun --script package --merge lualibs-extended.lua
```

Of course there is a make target for that:

```
make merge
```

will take care of assembling the packages from the files distributed with lua_{libs}.

For this to work, the syntax of the Lua file needs to be well-formed: files that should be merged must be included via a function `loadmodule()`. It doesn't matter if the function actually does something; a dummy will suffice. Also, the argument to `loadmodule()` must be wrapped in parentheses. This rule is quite convenient, actually, since it allows excluding files from the merge while still using `loadmodule()` consistently.

```
...
loadmodule("my-lua-file.lua") -- <= will be merged
loadmodule('my-2nd-file.lua') -- <= will be merged
loadmodule "my-3rd-file.lua" -- <= will be ignored
...
```

II lua_{libs}.lua

```

1 lualibs = lualibs or { }
2
3 lualibs.module_info = {
4   name      = "lualibs",
5   version   = 2.64,
6   date      = "2019-02-14",
7   description = "ConTeXt Lua standard libraries.",
8   author    = "Hans Hagen, PRAGMA-ADE, Hasselt NL & Elie Roux & Philipp Gesang",
9   copyright  = "PRAGMA ADE / ConTeXt Development Team",
10  license   = "See ConTeXt's mreadme.pdf for the license",
11 }
12

```

The behavior of the lualibs can be configured to some extent.

- Based on the parameter `lualibs.prefer_merged`, the libraries can be loaded via the included merged packages or the individual files.
- Two classes of libraries are distinguished, mainly because of a similar distinction in ConTeXt, but also to make loading of the less fundamental functionality optional. While the “basic” collection is always loaded, the configuration setting `lualibs.load_extended` triggers inclusion of the extended collection.
- Verbosity can be increased via the `verbose` switch.

```

13
14 config      = config or { }
15 config.lualibs = config.lualibs or { }
16
17 if config.lualibs.prefer_merged ~= nil then
18   lualibs.prefer_merged = config.lualibs.prefer_merged
19 else
20   lualibs.prefer_merged = true
21 end
22
23 if config.lualibs.load_extended ~= nil then
24   lualibs.load_extended = config.lualibs.load_extended
25 else
26   lualibs.load_extended = true
27 end
28
29 if config.lualibs.verbose ~= nil then
30   config.lualibs.verbose = config.lualibs.verbose
31 else
32   config.lualibs.verbose = false
33 end
34

```

The lualibs may be loaded in scripts. To account for the different environment, fallbacks for the luatexbase facilities are installed.

```

35
36 local dofile      = dofile

```

```

37 local kpsefind_file = kpse.find_file
38 local stringformat = string.format
39 local texiowrite_nl = texio.write_nl
40
41 local find_file, error, warn, info
42 do
43   local _error, _warn, _info
44   if luatexbase and luatexbase.provides_module then
45     _error, _warn, _info = luatexbase.provides_module(lualibs.module_info)
46   else
47     _error, _warn, _info = texiowrite_nl, texiowrite_nl, texiowrite_nl
48   end
49
50   if lualibs.verbose then
51     error, warn, info = _error, _warn, _info
52   else
53     local dummylogger = function ( ) end
54     error, warn, info = _error, dummylogger, dummylogger
55   end
56   lualibs.error, lualibs.warn, lualibs.info = error, warn, info
57 end
58
59 if luatexbase and luatexbase.find_file then
60   find_file = luatexbase.find_file
61 else
62   kpse.set_program_name"luatex"
63   find_file = kpsefind_file
64 end
65

```

The lualibs load a merged package by default. In order to create one of these, the meta file that includes the libraries must satisfy certain assumptions mt_x-package makes about the coding style. Most important is that the functions that indicates which files to include must go by the name `loadmodule()`. For this reason we define a `loadmodule()` function as a wrapper around `dofile()`.

```

66
67 local loadmodule = loadmodule or function (name, t)
68   if not t then t = "library" end
69   local filepath = find_file(name, "lua")
70   if not filepath or filepath == "" then
71     warn(stringformat("Could not locate %s \"%s\".", t, name))
72     return false
73   end
74   dofile(filepath)
75   return true
76 end
77
78 lualibs.loadmodule = loadmodule
79

```

The separation of the “basic” from the “extended” sets coincides with the split into luat-bas.mkiv and luat-lib.mkiv.

```
80
81 if lualibs.basic_loaded      ~= true
82 or config.lualibs.force_reload == true
83 then
84   loadmodule"lualibs-basic.lua"
85   loadmodule"lualibs-compat.lua" --- restore stuff gone since v1.*
86 end
87
88 if lualibs.load_extended     == true
89 and lualibs.extended_loaded  ~= true
90 or config.lualibs.force_reload == true
91 then
92   loadmodule"lualibs-extended.lua"
93 end
94
95 --- This restores the default of loading everything should a package
96 --- have requested otherwise. Will be gone once there is a canonical
97 --- interface for parameterized loading of libraries.
98 config.lualibs.load_extended = true
99
100 -- vim:tw=71:sw=2:ts=2:expandtab
101
```

III lualibs-basic.lua

```
1 lualibs          = lualibs or { }
2 local info      = lualibs.info
3 local loadmodule = lualibs.loadmodule
4
5 local lualibs_basic_module = {
6   name      = "lualibs-basic",
7   version   = 2.64,
8   date      = "2019-02-14",
9   description = "ConTeXt Lua libraries -- basic collection.",
10  author    = "Hans Hagen, PRAGMA-ADE, Hasselt NL & Elie Roux & Philipp Gesang",
11  copyright  = "PRAGMA ADE / ConTeXt Development Team",
12  license    = "See ConTeXt's mreadme.pdf for the license",
13 }
14
15 local loaded = false --- track success of package loading
16
17 if lualibs.prefer_merged then
18   info"Loading merged package for collection "basic"."
19   loaded = loadmodule('lualibs-basic-merged.lua')
20 else
21   info"Ignoring merged packages."
```

```

22 info"Falling back to individual libraries from collection "basic"."
23 end
24

```

mtx-package expects the files to be included by loadmodule. If run on this file, it will create lualibs-basic-merged.lua from all the files mentioned in the next block.

```

25
26 if loaded == false then
27   loadmodule("lualibs-lua.lua")
28   loadmodule("lualibs-package.lua")
29   loadmodule("lualibs-lpeg.lua")
30   loadmodule("lualibs-function.lua")
31   loadmodule("lualibs-string.lua")
32   loadmodule("lualibs-table.lua")
33   loadmodule("lualibs-boolean.lua")
34   loadmodule("lualibs-number.lua")
35   loadmodule("lualibs-math.lua")
36   loadmodule("lualibs-io.lua")
37   loadmodule("lualibs-os.lua")
38   loadmodule("lualibs-file.lua")
39   loadmodule("lualibs-gzip.lua")
40   loadmodule("lualibs-md5.lua")
41   loadmodule("lualibs-dir.lua")
42   loadmodule("lualibs-unicode.lua")
43   loadmodule("lualibs-url.lua")
44   loadmodule("lualibs-set.lua")
45 end
46
47 lualibs.basic_loaded = true
48 -- vim:tw=71:sw=2:ts=2:expandtab
49

```

IV lualibs-extended.lua

```

1 lualibs = lualibs or { }
2

```

Loading the *extended* set requires a tad more effort, but it's well invested.

Since we only want the functionality, we have to simulate parts of a running ConTeXt environment, above all logging, that some of the more involved libraries cannot be loaded without. Also, one utility file cannot be packaged because it returns a table which would preclude loading of later code. Thus, we remove it from the natural loading chain (it is not critical) and append it at the end.

```

3
4 local lualibs_extended_module = {
5   name      = "lualibs-extended",
6   version   = 2.64,
7   date      = "2019-02-14",
8   description = "ConTeXt Lua libraries -- extended collection.",

```



```

9 author      = "Hans Hagen, PRAGMA-ADE, Hasselt NL & Elie Roux & Philipp Gesang",
10 copyright   = "PRAGMA ADE / ConTeXt Development Team",
11 license     = "See ConTeXt's mreadme.pdf for the license",
12 }
13
14
15 local stringformat = string.format
16 local loadmodule   = luaLlibs.loadmodule
17 local texiowrite   = texio.write
18 local texiowrite_nl = texio.write_nl
19

```

Here we define some functions that fake the elaborate logging/tracking mechanism Con-
text provides.

```

20
21 local error, logger, mklog
22 if luatexbase and luatexbase.provides_module then
23   -- TODO test how those work out when running tex
24   local __error, ___, __logger =
25     luatexbase.provides_module(luaLlibs_extended_module)
26   error = __error
27   logger = __logger
28   mklog = function ( ) return logger end
29 else
30   mklog = function (t)
31     local prefix = stringformat("[%s] ", t)
32     return function (...)
33       texiowrite_nl(prefix)
34       texiowrite (stringformat(...))
35     end
36   end
37   error = mklog"ERROR"
38   logger = mklog"INFO"
39 end
40
41 local info = luaLlibs.info
42

```

We temporarily put our own global table in place and restore whatever we overloaded afterwards.

ConTeXt modules each have a custom logging mechanism that can be enabled for debugging. In order to fake the presence of this facility we need to define at least the function `logs.reporter`. For now it's sufficient to make it a reference to `mklog` as defined above.

```

43
44 local dummy_function = function ( ) end
45 local newline        = function ( ) texiowrite_nl"" end
46
47 local fake_logs = function (name)
48   return {

```

```

49     name      = name,
50     enable    = dummy_function,
51     disable   = dummy_function,
52     reporter  = mklog,
53     newline   = newline
54 }
55 end
56
57 local fake_trackers = function (name)
58     return {
59         name      = name,
60         enable    = dummy_function,
61         disable   = dummy_function,
62         register  = mklog,
63         newline   = newline,
64     }
65 end
66
67 local backup_store = { }
68
69 local fake_context = function ( )
70     if logs      then backup_store.logs      = logs      end
71     if trackers then backup_store.trackers = trackers end
72     logs        = fake_logs"logs"
73     trackers    = fake_trackers"trackers"
74 end
75
76

```

Restore a backed up logger if appropriate.

```

77 local unfake_context = function ( )
78     if backup_store then
79         local bl, bt = backup_store.logs, backup_store.trackers
80         if bl      then logs      = bl      end
81         if bt      then trackers = bt      end
82     end
83 end
84
85 fake_context()
86
87 local loaded = false
88 if lualibs.prefer_merged then
89     info"Loading merged package for collection "extended"."
90     loaded = loadmodule('lualibs-extended-merged.lua')
91 else
92     info"Ignoring merged packages."
93     info"Falling back to individual libraries from collection "extended"."
94 end
95
96 if loaded == false then

```

```

97 loadmodule("luautils-util-str.lua")--- string formatters (fast)
98 loadmodule("luautils-util-fil.lua")--- extra file helpers
99 loadmodule("luautils-util-tab.lua")--- extended table operations
100 loadmodule("luautils-util-sto.lua")--- storage (hash allocation)
101 -----("luautils-util-pck.lua")---!packers; necessary?
102 -----("luautils-util-seq.lua")---!sequencers (function chaining)
103 -----("luautils-util-mrg.lua")---!only relevant in mtx-package
104 loadmodule("luautils-util-prs.lua")--- miscellaneous parsers; cool. cool cool cool
105 -----("luautils-util-fmt.lua")---!column formatter (rarely used)
106 loadmodule("luautils-util-dim.lua")--- conversions between dimensions
107 loadmodule("luautils-util-jsn.lua")--- JSON parser
108
109 -----("luautils-trac-set.lua")---!generalization of trackers
110 -----("luautils-trac-log.lua")---!logging
111 loadmodule("luautils-trac-inf.lua")--- timing/statistics
112 loadmodule("luautils-util-lua.lua")--- operations on lua bytecode
113 loadmodule("luautils-util-deb.lua")--- extra debugging
114 loadmodule("luautils-util-tpl.lua")--- templating
115 loadmodule("luautils-util-sta.lua")--- stacker (for writing pdf)
116 end
117
118 unfake_context() --- TODO check if this works at runtime
119
120 luautils.extended_loaded = true
121 -- vim:tw=71:sw=2:ts=2:expandtab
122

```