The luatexbase-loader package

Manuel Pégourié-Gonnard
mpg@elzevir.fr

Élie Roux
elie.roux@telecom-bretagne.eu

v0.2 2010-05-12

Abstract

Lua modules are loaded using the `require()` function which, similarly to TEX's `\input`, takes care of locating the file and load it, but also makes a few supplementary checks, for example to avoid loading the same module twice. This package adapts the way the files are searched in order to accommodate the TDS as well as usual Lua naming conventions.

For higher-level functions related to Lua modules, see luatexbase-modutils, which also loads the present package.

1 Documentation

Starting with LuaTEX 0.45.0, `require()` uses Kpathsea for file searching when the library is initialised (which is always the case in TEX mode, unless explicitly disabled by the user). However, it does not respect the Lua convention that `require("foo.bar")` should look for `foo/bar.lua`. [1] This package implements such behaviour.

More precisely, it implements a new kpse searcher that looks for file `foo/bar` using Kpathsea with the format `lua` (that is, search along `LUAINPUTS` and try the following extensions: `.luc`, `.luctex`, `.texluc`, `.lua`, `.luatex`, `.texlua`). If this search fails, it falls back to `foo.bar` (along the same path with the same extensions).

Also, older versions of LuaTEX, such as 0.25.4 (TEX Live 2008), don't know about the `lua` format for kpse searching. So, an emulator for this function is provided. The emulator is not perfect, in particular it may find more results than the normal `lua` format search.[2] In order to ensure more homogeneous results across versions, this emulator is used as a fall-back when the real `lua` format search doesn't find any result.

Finally, a combined version of this new kpse searcher and the original function at `package.loaders[2]` (using first the new loader, then the old one if the new doesn't return any result) is installed as `package.loaders[2]`.

2 Implementation

2.1 TEX package

[1] Support for that has been added in rev 3558 of LuaTEX, currently unreleased but probably part of LuaTEX 0.54.
[2] An may also fail to find the file in particular cases, see comments in the implementation for details.

```
⟨*texpackage⟩
```

2.1.1 Preliminaries

Reload protection, especially for Plain TEX.

```
                \csname lltxb@loader@loaded\endcsname
\expandafter\let\csname lltxb@loader@loaded\endcsname\endinput
```

Catcode defenses.

```
\begingroup
  \catcode123 1 % {
  \catcode125 2 % }
  \catcode 35 6 % #
  \toks0{}%
  \def\x{}%
  \def\y#1 #2 {%
    \toks0\expandafter{\the\toks0 \catcode#1 \the\catcode#1}%
    \edef\x{\x \catcode#1 #2}}%
  \y 123 1  % {
  \y 125 2  % }
  \y  35 6  % #
  \y  10 12 % ^^J
  \y  34 12 % "
  \y  36 3  % $ $
  \y  39 12 % ’
  \y  40 12 % (
  \y  41 12 % )
  \y  42 12 % *
  \y  43 12 % +
  \y  44 12 % ,
  \y  45 12 % -
  \y  46 12 % .
  \y  47 12 % /
  \y  60 12 % <
  \y  61 12 % =
  \y  64 11 % @ (letter)
  \y  62 12 % >
  \y  95 12 % _ (other)
  \y  96 12 % ‘
  \edef\y#1{\endgroup\edef#1{\the\toks0\relax}\x}%
\expandafter\y\csname lltxb@loader@AtEnd\endcsname
```

Package declaration.

```
\begingroup
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1[#2]{\immediate\write16{Package: #1 #2}}
  \else
    \let\x\ProvidesPackage
  \fi
\expandafter\endgroup
\x{luatexbase-loader}[2010/05/12 v0.2 Lua module loader for LuaTeX]
```

Make sure LuaTeX is used.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input ifluatex.sty
\else
  \RequirePackage{ifluatex}
\fi
\ifluatex\else
  \begingroup
    \expandafter\ifx\csname PackageWarningNoLine\endcsname\relax
      \def\x#1#2{\begingroup\newlinechar10
        \immediate\write16{Package #1 warning: #2}\endgroup}
    \else
      \let\x\PackageWarningNoLine
    \fi
  \expandafter\endgroup
  \x{luatexbase-loader}{LuaTeX is required for this package. Aborting.}
  \lltxb@loader@AtEnd
  \expandafter\endinput
\fi
```

2.1.2 Main content

First load luatexbase-compat.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input luatexbase-compat.sty
\else
  \RequirePackage{luatexbase-compat}
\fi
```

Load the supporting Lua module. This one doesn't follow the usual naming conventions, since it won't be loaded with the usual functions for obvious bootstraping reasons.

```
\luatexbase@directlua{%
  local file = "luatexbase.loader.lua"
  local path = assert(kpse.find_file(file, 'tex'),
    "File '"..file.."' no found")
  texio.write_nl("("..path..")")
  dofile(path)}
```

That's all, folks!

```
\lltxb@loader@AtEnd
⟨/texpackage⟩
```

2.2 Lua module

```
⟨*luamodule⟩
module('luatexbase', package.seeall)
```

Emulate (approximatively) kpse's lua format search. More precisely, combine the search path of `texmfscripts` and `tex` in order to approximate `LUAINPUTS`. But we need to handle suffixes ourselves.

`lua_suffixes` is taken verbatim from Kpathsea's source (`tex-file.c`, constant `LUA_SUFFIXES`),[3].

```
local lua_suffixes = {
  ".luc", ".luctex", ".texluc", ".lua", ".luatex", ".texlua",
}
```

Auxiliary function for suffixes: says if `suffix` is a suffix of `name`.

```
local function ends_with(suffix, name)
    return name:sub(-suffix:len()) == suffix
end
```

The search function first builds the list of filenames to be search. For the lua format, kpse always adds a suffix if no (known) suffix is present, so we do the same.

```
function find_file_lua_emul(name)
  local search_list = {}
  for _, suffix in ipairs(lua_suffixes) do
    if ends_with(suffix, name) then
      search_list = { name }
      break
    else
      table.insert(search_list, name..suffix)
    end
  end
```

Now look for each file in this list.

```
  for _, search_name in ipairs(search_list) do
    local f = kpse.find_file(search_name, 'texmfscripts')
      or kpse.find_file(search_name, 'tex')
```

There is a problem with using the `tex` search format: kpse will try to add suffixes from the `TEX_SUFFIXES` list, which may lead to problems if a file like ⟨*name*⟩`.lua.tex` exists. We prevent that by checking if the file found ends with the required name. So ⟨*name*⟩`.lua` will still be hidden by ⟨*name*⟩`.lua.tex` but it seems less bad not to find it than to return an incorrect result.

```
    if f and ends_with(search_name, f) then
      return f
    end
  end
end
```

If lua search format is available, use it with emulation as a fall-back, or just use emulation.

```
local find_file_lua
if pcall('kpse.find_file', 'dummy', 'lua') then
  find_file_lua = function (name)
    return kpse.find_file(name, 'lua') or find_file_lua_emul(name)
  end
else
  find_file_lua = function (name)
    return find_file_lua_emul(name)
```

[3] Unchanged since 2007-07-06, last checked 2010-05-10.

```
  end
end
```

Find the full path corresponding to a module name.

```
local function find_module_file(mod)
  return find_file_lua(mod:gsub('%.', '/'), 'lua')
    or find_file_lua(mod, 'lua')
end
```

Combined searcher, using primarily the new kpse searcher, and the original as a fall-back.

```
local package_loader_two = package.loaders[2]
local function load_module(mod)
  local file = find_module_file(mod)
  if not file then
    local msg = "\n\t[luatexbase.loader] Search failed"
    local ret = package_loader_two(mod)
    if type(ret) == 'string' then
      return msg..ret
    elseif type(ret) == 'nil' then
      return msg
    else
      return ret
    end
  end
  local loader, error = loadfile(file)
  if not loader then
    return "\n\t[luatexbase.loader] Loading error:\n\t"..error
  end
  texio.write_nl("("..file..")")
  return loader
end
```

Finally install this combined loader as loader 2.

```
package.loaders[2] = load_module
```

⟨/luamodule⟩

3 Test files

A dummy lua file for tests.

⟨*testdummy⟩

```
return true
```

⟨/testdummy⟩

Check that the package loads properly, under both LaTeX and Plain TeX, and load a dummy module in the current diretory.

⟨testplain⟩`\input luatexbase-loader.sty`

⟨testlatex⟩`\RequirePackage{luatexbase-loader}`

⟨*testplain, testlatex⟩

```
\catcode64 11
\luatexbase@directlua{require "test-loader"}
\luatexbase@directlua{require "test-loader.sub"}
```

149 ⟨/testplain, testlatex⟩
150 ⟨testplain⟩\bye
151 ⟨testlatex⟩\stop