The luatexbase-loader package

Heiko Oberdiek (primary author of luatex)
Élie Roux, Manuel Pégourié-Gonnard, Philipp Gesang*
https://github.com/lualatex/luatexbase
lualatex-dev@tug.org

v0.5 2013-04-13

Abstract

Lua modules are loaded using the `require()` function which, similarly to TEX's `\input`, takes care of locating the file to be loaded. 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, `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 did not respect the Lua convention that `require("foo.bar")` should look for `foo/bar.lua` until version 0.60. LuaTEX 0.74 ships with Lua 5.2 that has a different loading system.

The aim of this package is to have a coherent behaviour between versions of LuaTeX, and to get the loaded file's name printed in the output (LATEXstyle). The first versions did ensure backward compatibilty to LuaTEX 0.25 but as LuaTEX development is quite fast, this version supports only LuaTEX 0.45 and higher.

More precisely, when asked for `foo.bar` (or `foo.bar.lua`), it first looks for file `foo/bar` using Kpathsea with the format `lua`, and then for `foo.bar`, removing the possible extension.

2 Implementation

2.1 TEX package

1 ⟨*texpackage⟩

2.1.1 Preliminaries

Catcode defenses and reload protection.

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax% = and space
3   \catcode123 1 % {
4   \catcode125 2 % }
```

*See "History" in luatexbase.pdf for details.

```
  \catcode 35 6 % #
  \toks0\expandafter{\expandafter\endlinechar\the\endlinechar}%
  \edef\x{\endlinechar13}%
  \def\y#1 #2 {%
    \toks0\expandafter{\the\toks0 \catcode#1 \the\catcode#1}%
    \edef\x{\x \catcode#1 #2}}%
  \y  13  5 % carriage return
  \y  61 12 % =
  \y  32 10 % space
  \y 123  1 % {
  \y 125  2 % }
  \y  35  6 % #
  \y  64 11 % @ (letter)
  \y  10 12 % new line ^^J
  \y  34 12 % "
  \y  39 12 % '
  \y  40 12 % (
  \y  41 12 % )
  \y  44 12 % ,
  \y  45 12 % -
  \y  46 12 % .
  \y  47 12 % /
  \y  58 12 % :
  \y  91 12 % [
  \y  93 12 % ]
  \y  94  7 % ^
  \y  95  8 % _
  \y  96 12 % `
  \toks0\expandafter{\the\toks0 \relax\noexpand\endinput}%
  \edef\y#1{\noexpand\expandafter\endgroup%
    \noexpand\ifx#1\relax \edef#1{\the\toks0}\x\relax%
    \noexpand\else \noexpand\expandafter\noexpand\endinput%
    \noexpand\fi}%
\expandafter\y\csname luatexbase@loader@sty@endinput\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}[2013/04/13 v0.5 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 PackageError\endcsname\relax
      \def\x#1#2#3{\begingroup \newlinechar10
        \errhelp{#3}\errmessage{Package #1 error: #2}\endgroup}
    \else
      \let\x\PackageError
    \fi
  \expandafter\endgroup
  \x{luatexbase-loader}{LuaTeX is required for this package. Aborting.}{%
    This package can only be used with the LuaTeX engine^^J%
    (command 'lualatex' or 'luatex').^^J%
    Package loading has been stopped to prevent additional errors.}
  \expandafter\luatexbase@loader@sty@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, 'lua'),
    "File '"..file.."' not found")
  texio.write_nl("("..path..")")
  dofile(path)}
```

That's all, folks!

```
\luatexbase@loader@sty@endinput%
```
⟨/texpackage⟩

2.2 Lua module

⟨*luamodule⟩

```
luatexbase          = luatexbase or { }
local luatexbase    = luatexbase
```

Just in case it's called from a TEXLua script...

```
kpse.set_program_name("luatex")
```

In LATEX, it's traditional to print the included file paths. We don't want to do this for scripts using texlua... Currently there is no perfect check of texlua vs. luatex, so we check for the token table.

```
local print_included_path = false
if token then
  print_included_path = true
end
```

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`).[1]

```
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
```

Auxiliary function for suffixes: returns the basename of a file if it end by one of the suffixes.

```
local function basename(name)
  for _, suffix in ipairs(lua_suffixes) do
    if ends_with(suffix, name) then
      return name:sub(1, -(suffix:len()+1))
    end
  end
  return name
end
```

The main function, emulating the behaviour of `packages.searchers[2]`, with a small improvement that eliminates the possible extension.

```
local function find_module_file(mod)
  local compat = basename(mod):gsub('%.', '/')
  return kpse.find_file(compat, 'lua') or kpse.find_file(mod, 'lua')
end
```

Combined searcher, using primarily the new kpse searcher, and the original as a fall-back. Starting from LuaTEX 0.75, Lua 5.2 is used. Among the changes, `package.loaders` is renamed as `package.searchers`.

The main improvement is thus the printing of the filename in the output.

```
local package_loader_two
if not package.searchers then
  package.searchers = package.loaders
end
package_loader_two = package.searchers[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)
```

[1] Last checked 2013-04-12.

```
  if not loader then
    return "\n\t[luatexbase.loader] Loading error:\n\t"..error
  end
  if print_included_path then
    texio.write_nl("("..file..")")
  end
  return loader
end
```

Finally install this combined loader as loader 2.

```
package.searchers[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"}
```
⟨/testplain, testlatex⟩
⟨testplain⟩`\bye`
⟨testlatex⟩`\stop`