The luatexbase-compat package

Manuel Pégourié-Gonnard & Élie Roux
Support: lualatex-dev@tug.org

2010/10/10 v0.3

Abstract

The LuaTEX manual is very clear: everything may change. This package provides tools to help package writers deal with the changes. It helps supporting LuaTEX versions down to 0.25.4, and is regularly tested with LuaTEX 0.40.6 (TEX Live 2009) and from trunk.

Contents

1 Documentation

Three problems are currently addressed by this package: changes in the syntax of `\directlua`, version information, and variable policies for primitives activation and naming (in LuaTEX itself as well as in TEX Live).

Older versions of LuaTEX used to support multiple Lua states. A number was mandatory with `\directlua` in order to specify the Lua state to be used. Later, support for multiple Lua states was removed and the old syntax resulted in a warning. Now (LuaTEX 0.50), `\directlua` again accepts a number after `\directlua`, but with a different meaning (see the LuaTEX manual for details).

This package provides a macro `\luatexbase@directlua` that expands to `\directlua0` on LuaTEX 0.35 and lower (where the number is mandatory), and to `\directlua` otherwise. It is a macro in both case so that the number of expansion steps remains constant.

Current versions of LuaTEX make the version available directly from Lua as `tex.luatexversion` and `tex.luatexrevision`. However, older versions (such as 0.25.4) didn't, which makes it particularly uneasy to test the version from within Lua. The present package makes this information available as `luatexbase.luatexversion` and `luatexbase.luatexrevision`.

Starting with LuaTeX 0.39.0, the only primitives available in IniTeX mode are the basic primitives from TeX82 and `\directlua`. All other primitives are hidden by default and have to be activated using a Lua function. In TeX Live 2009 (LuaTeX 0.40.6), the following arrangement has been made in order to try preserving usability while avoiding name clashes in the LaTeX world: in LaTeX-based formats, all pdfTeX primitives are enabled with their normal name, but the primitives specific to LuaTeX are enabled with the `luatex` prefix.[1] In Plain based formats however, all the primitives are enabled with their natural name, but are also provided with the same name as in LaTeX-based formats in order to help writing generic packages.

So, starting with TeX Live 2009, the situation is clear: the prefixed version of the LuaTeX primitives is always available. But in earlier versions (TeX Live 2008, LuaTeX 0.25.4) those primitives were available only with their natural names. Also, it is theoretically possible, however unlikely, that the prefixed primitives are not available for some reason.

`\luatexbase@ensure@primitive{`⟨*name*⟩`}`

The tool provided to deal with that is `\luatexbase@ensure@primitive`, whose argument is a primitive name (without a leading backslash nor any `luatex` prefix, eg just `{latelua}`). It makes sure that the primitive gets available as `\luatex`⟨*name*⟩.

Warning. In particular circumstances, this macro may fail silently for primitives whose natural name starts with `luatex`, hence such primitives shouldn't be used as arguments. This is actually not a problem, since the only three such primitives are `\luatexversion`, `\luatexrevision` and `\luatexdatestamp`. The first two are already activated by ifluatex which is loaded by this package, so you don't need to activated them yourself. The third should never be used in production according to the LuaTeX manual.

Remark. If you only aim at compatibility down to TeX Live 2009 (LuaTeX 0.40.6), then you can simply use the primitives with their prefixed name (except for `\directlua` which never needs a prefix). If you want extra security and/or compatibility down to TeX Live 2008 (LuaTeX 0.25.4) then you should use `\luatebase@ensure@primitive` for each primitive you intend to use (except `\directlua` again).

This package doesn't try to activate every primitive, since it would require and extensive list of primitives for each version of LuaTeX, so it seems simpler to leave that burden on package writers.

2 Implementation

```
⟨*texpackage⟩
```

2.1 Preliminaries

Reload protection, especially for Plain TeX.

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

Catcode defenses.

```
\begingroup
  \catcode123 1 % {
  \catcode125 2 % }
  \catcode 35 6 % #
  \toks0{}%
```

[1] The prefix is dropped for primitives whose name already starts with `luatex`.

```
  \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@compat@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-compat}[2010/10/10 v0.3 Compatibility tools 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-attr}{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.}
  \lltxb@compat@AtEnd
  \expandafter\endinput
\fi
```

2.2 \directlua abstraction

Define `\luatexbase@directlua` to be either `\directlua0` or `\directlua`, depending on the version of LuaTEX.

```
\begingroup
\expandafter\ifx\csname newcommand\endcsname\relax
  \toks0{\long\def\luatexbase@directlua}%
\else
  \toks0{\newcommand\luatexbase@directlua}%
\fi
\ifnum\luatexversion<36
  \toks0\expandafter{\the\toks0{\directlua0}}%
\else
  \toks0\expandafter{\the\toks0{\directlua}}%
\fi
\expandafter\endgroup
\the\toks0
```

2.3 Version information

Make `\luatexversion` and `\luatexrevision` available from Lua.

```
\luatexbase@directlua{%
  luatexbase = luatexbase or {}
  luatexbase.luatexversion = \number\luatexversion\space
  luatexbase.luatexrevision = \number\luatexrevision\space}
```

2.4 Primitives

Try reasonably hard to activate a primitive. First, check if it is already activated an do nothing in this case.

```
\begingroup
\expandafter\ifx\csname newcommand\endcsname\relax
  \toks0{\def\luatexbase@ensure@primitive#1}
\else
  \toks0{\newcommand*\luatexbase@ensure@primitive[1]}
\fi
\toks2{}\def\x#1{\toks2\expandafter{\the\toks2 #1}}
\x{%
  \ifcsname luatex#1\endcsname \else}
\ifnum\luatexversion<37\relax
```

`tex.enableprimitives()` not available. If the unprefixed primitive is undefined, issue an error.

```
  \x{%
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname #1\endcsname\relax}
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname PackageError\endcsname\relax
    \x{%
      \errmessage{%
        Package luatexbase-compat error: failed to enable ‘#1’.}}
  \else
    \x{%
      \PackageError{luatexbase-compat}{%
        Package luatexbase-compat error: failed to enable ‘#1’.}{}}
  \fi
  \x{%
    \else}
```

Use the unprefixed primitive to define the prefixed version.

```
  \x{%
      \expandafter\let\csname luatex#1\expandafter\endcsname
      \csname#1\endcsname
    \fi}
\else
```

`tex.enableprimitives()` available, use it.

```
  \x{%
    \luatexbase@directlua{tex.enableprimitives('luatex', {'#1'})}}
\fi
\x{%
  \fi}
\toks0\expandafter{\the\toks0\expandafter{\the\toks2}}
\expandafter\endgroup
\the\toks0
```

That’s all folks!

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

3 Test files

Test fils for Plain and LaTeX

```
⟨testplain⟩\input luatexbase-compat.sty
⟨testlatex⟩\RequirePackage{luatexbase-compat}
⟨*testplain, testlatex⟩
\catcode64 11
\luatexbase@directlua{local answer = 42}
\luatexbase@ensure@primitive{primitive}
\luatexprimitive\relax
\luatexbase@directlua{assert(type(luatexbase.luatexversion) == 'number')}
\luatexbase@directlua{assert(type(luatexbase.luatexrevision) == 'number')}
⟨/testplain, testlatex⟩
⟨testplain⟩\bye
⟨testlatex⟩\stop
```