LuaTEX Lua modules on Linux

Taco Hoekwater

How to use the dynamic Lua module loading abilities in LuaTEX under Linux or similar systems.

1. Introduction

First, I should warn you that dynamic loading of modules is not quite trivial. It is simple enough if everything works as expected, but lots of problems can come up and if you are not a programmer it can be extremely confusing.
It is hoped that eventually some knowledgeable people will create ready-to-use packages that are then made available via the normal TEX distributions like TEX Live, MikTEX, or the ConTEXt minimals. But for now, you have to do it on your own (or find a programming friend to do it for you).
Since it is a popular database back-end and LuaTEX presents a decent interface for typesetting from a database, I will use the luasqlite3 module as an example. The latest version of this module is here: http://lua.sqlite.org/index.cgi/index

2. Building a module

On any Unix-like platform, you will first have to compile the module. Here is a step by step guide for Linux (I don't know how to compile modules for Windows or MacOSX, sorry), which is what I use, but for most Unix-like system the procedure is very close, if not identical:

- Make sure you have both the `lua51` and `sqlite3` development packages installed. You need the development version of the packages because the module that we will build will be linked against those libraries. You can use your normal package manager for that, but the names of the packages may vary a little: the exact names on my Linux distribution were `liblua-devel` (version 5.1.4) and `libsqlite3-devel` (version 3.7.3).
- Then, download the module source from the URL given above, and unpack the zip file somewhere where you have write access. This will give you a directory `lsqlite3_svn08` or similar.
- Move into the `lsqlite3_svn08` directory, and type `make`. With some luck, this is good enough, and you will now have `lsqlite3.so`. If `make` fails, it is likely because you have to adjust some of the variables in the top of the `Makefile`. Open `Makefile` in an editor and have a look. Usually, there are a few helpful comments included explaining what you have to modify. Of course, if you cannot figure out what to do from that, you can always ask on the luatex mailing list (`luatex@tug.org`).
  Also, usually there is a file named `README` or `INSTALL` that can contain valuable hints.
- To verify that the created `lsqlite3.so` file is actually OK, type `make test`. The output should end with:

```
#### Test Suite finished.

479 Assertions checked. All
                  Tests passed!
```

  All the hard parts have now been done, we just have to copy the file to a more useful location.

3. Installation

3.1 for texlua

If LuaTEX runs as `texlua` (a.k.a. Lua interpreter mode), it will search for `lsqlite3.so` in exactly the same way as the standalone Lua does.
So, for that case, a simple `make install` will do the trick (you will probably need sudo rights). This will install the module in the correct system-wide spot and both `lua` and `texlua` will be able to find it.
Testing the standalone `texlua` is a simple case of, after `make install`, executing `make clean` to remove the generated files in the local directory and then using `texlua` to run the test suite instead of `lua`:

```
[... lsqlite3_svn08]$
              texlua tests-sqlite3.lua
```

3.2 for typesetting

Having successfully created the shared library (`.so`) file, now we also have to put it where LuaTEX can find it while typesetting.
To this end, LuaTEX uses a new kpathsea file type created specifically for this purpose: `clua`. This file type searches for files with extension `.dll` and `.so`. The `texmf.cnf` variable for this new file type is `CLUAINPUTS`, and by default it has this value:

```
CLUAINPUTS = .:$SELFAUTOLOC/lib/
            {$progname,$engine,}/lua//
```

This path is a bit odd because it requires a TDS subtree below the binaries directory, but the architecture has to be in the path somewhere and the simplest way to do that is to search below the binaries directory only.
In my case, the LuaTEX executable lives in `/opt/tex/texmf-linux/bin/` which replaces `$SELFAUTOLOC`.We will add the `luatex` path part as well as that is a nice thing to do (this replaces `$engine`), so the `lsqlite3.so` file should go here:

```
/opt/tex/texmf-linux/bin/lib/luatex/lua
```

Just copy the file there manually (again, you may need sudo rights).

To test, create a minimal input file with the name `t-sqlite3.tex` containing this:

```
\directlua{dofile
               ('tests-sqlite3.lua');}
\bye
```

and run it from that same directory:

```
[... lsqlite3_svn08]$
                  luatex t-sqlite3.tex
```

You should get the same output again.