This Way

ConTEXt magazine #12
August 2007

MkIV Goes Beta

Hans Hagen
PRAGMA ADE

This document describes (shortly) how to get the ConTEXt MkIV beta up and running. This variant is LuaTEX aware and is downward compatible with MkII. Eventually new functionality will be brought in. The main release will happen about a year from now.

1 Introduction

The history of MkIV is described elsewhere, as is the new low level functionality of LuaTEX, the new kid on the block for which MkIV is written.

The average user may not notice the difference between MkIV and MkII, but a glance at the log file may confirm that there are differences. These will be discussed at the mailing list and on the Wiki. You may also notice that the LuaTEX–MkIV pair runs somewhat slower than pdfTEX–MkII. This has to do with the fact that font handling is now done in Lua and advanced new features demand extensive processing of the internal node lists. Some parts of the code are rather well optimized, for instance file handling is on the average faster than in pdfTEX. Instead of macro based utf handling, we now have native utf support.

If you look at the files with suffix `mkiv` and `lua` you will see quite some experimental code. Don't worry, not all code is enabled. For instance attribute handling still has to be integrated. Also, font feature support will be improved, completed and sped up. Currently OpenType features are bound to a font instance but as soon as we can dynamically switch features, less font instances are needed which will gain us some speed.

Eventually we will have a set of rather generic modules which makes it possible to have most (Lua) based functionality also available in for instance Plain TEX.

2 Installation

First of all, we advise you to define an environment variable `TEXMFCACHE` which points to your favourite temporary path. On my machine this is

```
set TEXMFCACHE=c:\temp
```

Of course you need a recent version of LuaTEX. You need to put the binary in your path. The pool file is includes in the binary so you don't need to worry about that.

- copy luatex(.exe) to the binary path
- copy luatex(.exe) to the binary path as texlua(.exe)

Next you need to install a copy of ConTEXt:

- unzip cont-tmf.zip in your `texmf-local` tree

Since you update, make sure to remake the pdfTEX and XETEX formats:

```
texexec --make --all --pdftex
texexec --make --all --xetex
```

In principle you can also say:

```
texexec --make --all --luatex
```

In this case TEXexec will delegate format generation to luatools, so we have to install this program first:

- MS Windows: copy both `luatools.cmd` and `luatools.lua` to the binary path
- Unix: copy `luatools.lua` to the binary path as `luatools` (no suffix)

If you use Unix, you may need to apply:

```
tounix luatools
chmod 755 luatools
```

The scripts can be found in

```
texmf-local/scripts/context/lua
```

It makes sense to install `mtxrun.lua` in the same way. This is a replacement for texmfstart that eventually will replace that script.

Now you can generate formats with TEXexec or you can run luatools directly:

```
luatools --generate
luatools --ini --compile --verbose cont-en
```

From the console you may deduce that we put file databases as well as formats in the temporary path in a tree specific location. When TEXLua is used, luatools will scan the disk itself, otherwise it will use the `ls-r` database. Best let luatools do the work. For the moment we use the good old `texmf.cnf` file (although it is converted into a more suitable format) but this may change in the future.

In the cache path you will find files with the suffix `tma` (Lua code, normally a table) and `tmc` (the byte–compiled version of the former). In you runtime path you will find `tua` and `tuc` files. These are the counterparts (and to be replacements) of the utility files (multipass data).

In order to be able to start up LuaTEX needs a Lua file (unless in traditional mode) and luatools will handle this. When things fail, make sure that you have set the following variables:

```
TEXMFCNF=/tex/texmf-local/web2c
TEXMF={/tex/texmf,/tex/texmf-local}
```

We don't use any kpse code in MkIV so it may be that you need to help it a bit in locating the configuration file and trees. Don't forget the braces when your variable is a bit more complex! Of course you need values that make

sense at your system. When you generate the file databases you should get something reported that looks like:

```
LuaTools | loading t:/minimal/texmf-local/web2c/texmf.cnf
LuaTools | loading t:/minimal/texmf/web2c/texmf.cnf
LuaTools | preparing configuration in c:/temp/luatex-cache/context/d2...df/trees/a352eae69dd96910a5515cd7c9351551.tma
LuaTools | saving configuration in c:/temp/luatex-cache/context/d2...df/trees/a352eae69dd96910a5515cd7c9351551.tma
LuaTools | compiling configuration to c:/temp/luatex-cache/context/d2...df/trees/a352eae69dd96910a5515cd7c9351551.tmc
LuaTools | preparing configuration in c:/temp/luatex-cache/context/d2...df/trees/ce66215f31bd82750f81eeab38aab1d4.tma
LuaTools | saving configuration in c:/temp/luatex-cache/context/d2...df/trees/ce66215f31bd82750f81eeab38aab1d4.tma
LuaTools | compiling configuration to c:/temp/luatex-cache/context/d2...df/trees/ce66215f31bd82750f81eeab38aab1d4.tmc
LuaTools | locating list of t:/minimal/texmf-mswin
LuaTools | locating list of t:/minimal/texmf-project
LuaTools | locating list of t:/minimal/texmf-fonts
LuaTools | locating list of t:/minimal/texmf-local
LuaTools | locating list of t:/minimal/texmf-extra
LuaTools | locating list of t:/minimal/texmf
LuaTools | scanning path t:/minimal/texmf-mswin
LuaTools | 130 files found on 4 directories
LuaTools | scanning path t:/minimal/texmf-fonts
LuaTools | 532 files found on 45 directories
LuaTools | scanning path t:/minimal/texmf-local
LuaTools | 1474 files found on 138 directories
LuaTools | scanning path t:/minimal/texmf-extra
LuaTools | 19 files found on 17 directories
LuaTools | scanning path t:/minimal/texmf
LuaTools | 5167 files found on 197 directories
LuaTools | preparing files in c:/temp/luatex-cache/context/d2...df/trees/084a237b6d002dc35a14a188768c3a78.tma
LuaTools | saving files in c:/temp/luatex-cache/context/d2...df/trees/084a237b6d002dc35a14a188768c3a78.tma
LuaTools | compiling files to c:/temp/luatex-cache/context/d2...df/trees/084a237b6d002dc35a14a188768c3a78.tmc
LuaTools | preparing files in c:/temp/luatex-cache/context/d2...df/trees/9e7cc7cf83c346eca4d23466474dd0ee.tma
LuaTools | saving files in c:/temp/luatex-cache/context/d2...df/trees/9e7cc7cf83c346eca4d23466474dd0ee.tma
LuaTools | compiling files to c:/temp/luatex-cache/context/d2...df/trees/9e7cc7cf83c346eca4d23466474dd0ee.tmc
LuaTools | preparing files in c:/temp/luatex-cache/context/d2...df/trees/8322b38686a2adbfac823764135bdbd8.tma
LuaTools | saving files in c:/temp/luatex-cache/context/d2...df/trees/8322b38686a2adbfac823764135bdbd8.tma
LuaTools | compiling files to c:/temp/luatex-cache/context/d2...df/trees/8322b38686a2adbfac823764135bdbd8.tmc
LuaTools | preparing files in c:/temp/luatex-cache/context/d2...df/trees/af041a3f9d1037b124a0a45c60cbe1f1.tma
LuaTools | saving files in c:/temp/luatex-cache/context/d2...df/trees/af041a3f9d1037b124a0a45c60cbe1f1.tma
LuaTools | compiling files to c:/temp/luatex-cache/context/d2...df/trees/af041a3f9d1037b124a0a45c60cbe1f1.tmc
LuaTools | preparing files in c:/temp/luatex-cache/context/d2...df/trees/68cf73c891416316857d593095f7c596.tma
LuaTools | saving files in c:/temp/luatex-cache/context/d2...df/trees/68cf73c891416316857d593095f7c596.tma
LuaTools | compiling files to c:/temp/luatex-cache/context/d2...df/trees/68cf73c891416316857d593095f7c596.tmc
LuaTools |
LuaTools | runtime: 0.965 seconds
```

When you want to test the databases, you can try:

```
luatools lmr12.afm
```

On my system this reports:

```
c:/data/develop/tex/texmf-fonts/fonts/data/e-foundry/latin-modern/lmr12.afm
```

3 Fonts

We assume that you have installed the open type versions of the Latin Modern and TEX Gyre fonts. If you have installed mtxrun you can test this with:

```
mtxrun --script fonts --list --pattern=pagella
```

This should give something like:

```
texgyrepagella-bold        TeXGyrePagella-Bold        texgyrepagella-bold.otf
texgyrepagella-bolditalic  TeXGyrePagella-BoldItalic  texgyrepagella-bolditalic.otf
texgyrepagella-italic      TeXGyrePagella-Italic      texgyrepagella-italic.otf
texgyrepagella-regular     TeXGyrePagella-Regular     texgyrepagella-regular.otf
```

These are the names that MkIV will recognize when you call for these fonts. When a requested name is not found, MkIV will generate a database with possible names. In that case it will report a log of found tree and system fonts. System font paths are specified with the `OSFONTDIR` environment variable or preferably in `texmf.cnf`.

```
report >> fontnames: reloading font database
report >> fontnames: identifying tree font files with suffix otf
report >> fontnames: 72 tree files identified, 234 hash entries added, runtime 0.153 seconds
report >> fontnames: identifying tree font files with suffix ttf
report >> fontnames: 1 tree files identified, 1 hash entries added, runtime 0.035 seconds
report >> fontnames: identifying tree font files with suffix ttc
report >> fontnames: 0 tree files identified, 0 hash entries added, runtime 0.035 seconds
report >> fontnames: identifying tree font files with suffix afm
report >> fontnames: 521 tree files identified, 964 hash entries added, runtime 0.186 seconds
report >> fontnames: identifying system font files with suffix otf
report >> fontnames: 0 system files identified, 0 hash entries added, runtime 0.039 seconds
report >> fontnames: identifying system font files with suffix ttf
report >> fontnames: 193 system files identified, 614 hash entries added, runtime 0.187 seconds
report >> fontnames: identifying system font files with suffix ttc
report >> fontnames: 10 system files identified, 98 hash entries added, runtime 0.076 seconds
report >> fontnames: identifying system font files with suffix afm
report >> fontnames: 0 system files identified, 0 hash entries added, runtime 0.029 seconds
report >> fontnames: done
```

Actually MkIV is rather tolerant in resolving font names. Because font names are often rather inconsistent (across the domain of names) we also support collapsed and stripped versions as well as combinations of main characteristics. Future versions may also use additional third party databases (given that they are available).

```
mtxrun --script fonts --list --all --pattern=pagella
```

```
texgyrepagella bold        TeXGyrePagella-Bold        texgyrepagella-bold.otf
texgyrepagella book        TeXGyrePagella-Regular     texgyrepagella-regular.otf
texgyrepagella-bold        TeXGyrePagella-Bold        texgyrepagella-bold.otf
texgyrepagella-bolditalic  TeXGyrePagella-BoldItalic  texgyrepagella-bolditalic.otf
texgyrepagella-italic      TeXGyrePagella-Italic      texgyrepagella-italic.otf
texgyrepagella-regular     TeXGyrePagella-Regular     texgyrepagella-regular.otf
texgyrepagellabold         TeXGyrePagella-Bold        texgyrepagella-bold.otf
texgyrepagellabolditalic   TeXGyrePagella-BoldItalic  texgyrepagella-bolditalic.otf
texgyrepagellabook         TeXGyrePagella-Regular     texgyrepagella-regular.otf
texgyrepagellaitalic       TeXGyrePagella-Italic      texgyrepagella-italic.otf
texgyrepagellaregular      TeXGyrePagella-Regular     texgyrepagella-regular.otf
```

However, we have adapted the existing typescripts so that they will choose an open type font automatically when you ask for a Palatino. A good test is the following file:

```
\starttext

\usetypescript[serif][palatino]

\startlines
{\definefontsynonym[Test][Palatino]      [features=smallcaps]\definedfont[Test] effe fietsen (caps)  }
{\definefontsynonym[Test][Palatino]      [features=default]  \definedfont[Test] effe fietsen (normal)}
{\definefontsynonym[Test][Palatino]                          \definedfont[Test] effe fietsen (normal)}
{\definefontsynonym[Test][Palatino-Caps][features=smallcaps]\definedfont[Test] effe fietsen (caps)  }
{\definefontsynonym[Test][Palatino-Caps][features=default]  \definedfont[Test] effe fietsen (normal)}
{\definefontsynonym[Test][Palatino-Caps]                    \definedfont[Test] effe fietsen (caps)  }
\stoplines

\stoptext
```

Don't forget the `--luatex` directive for TEXexec, or add the engine directive to the file:

```
% engine=luatex
```

Depending on what fonts are available, MkIV will use OpenType or Type1 instances. It will either use the information from the OpenType file or filter information from afm files. If none of these is present, it will look for an tfm file. What fonts are used can be see at the end of the log, This is what the log of this document shows:

```
systems : input load time - 0.088 seconds
systems : fonts load time - 2.105 seconds
systems : mps conversion time - 0.001 seconds
systems : node processing time - 0.221 second
systems : used config path - c:/data/develop/tex/texmf-mine/web2c
```

```
systems : used cache path - c:/temp/luatex-cache/context/36cfaf81cc343ab858e892f7ccbb5039
systems : lua memory usage - 170244849 bytes
systems : loaded fonts - cmtt10:tfm lmex10:tfm lmmi10:tfm lmmi12:tfm lmmi5:tfm lmmi7:tfm lmmi9:tfm
    lmroman10-bold*default:otf lmroman10-bolditalic*default:otf lmroman10-boldoblique*default:otf
    lmroman10-capsregular*default:otf lmroman10-italic*default:otf lmroman10-oblique*default:otf
    lmroman10-regular*default:otf lmroman12-bold*default:otf lmroman12-italic*default:otf
    lmroman12-oblique*default:otf lmroman12-regular*default:otf lmroman5-bold*default:otf
    lmroman5-regular*default:otf lmroman7-bold*default:otf lmroman7-regular*default:otf
    lmroman9-bold*default:otf lmroman9-italic*default:otf lmroman9-oblique*default:otf
    lmroman9-regular*default:otf lmsy10:tfm lmsy5:tfm lmsy7:tfm lmsy9:tfm
    lmtypewriter10-lightcondensed:otf lmtypewriter10-regular:otf msam10:tfm msam5:tfm msam7:tfm
    msbm10:tfm msbm5:tfm msbm7:tfm pxex:tfm pxmi:tfm pxr:tfm pxsy:tfm pxsya:tfm pxsyb:tfm rm-lmr10:tfm
    rm-lmr12:tfm rm-lmr5:tfm rm-lmr7:tfm rm-lmr9:tfm rpxmi:tfm rpxpplr:tfm rpxpplri:tfm rpxr:tfm
    texgyrepagella-bold*default:otf texgyrepagella-bolditalic*default:otf texgyrepagella-italic*default:otf
    texgyrepagella-regular*default:otf texgyrepagella-regular*smallcaps:otf
systems : lua modules/bytes - 68/3140217
systems : lua instances/bytes - 1/170253423
```

When a font is loaded the first time, it may take some extra time because a cached instance is generated.

This is no manual on fonts, nor on OpenType features. However, if you want to experiment with fonts, here is an indication of how more advanced things can be achieved:

```
\definefontfeature
  [arabtest]
  [mode=node,language=dflt,script=arab,
   liga=yes,dlig=yes,rlig=yes,calt=yes,clig=yes,
   init=yes,medi=yes,fina=yes,isol=yes,
   mkmk=yes,mark=yes,kern=yes,curs=yes]

\definefontsynonym
  [Arabic Typesetting]
  [arabtype]
  [features=arabtest]

\definefont
  [MyFont]
  [Arabic Typesetting]
```

After this, `\MyFont` should switch to Arabic rendering. Of course some more is needed, like:

```
\textdir TRT \pardir TRT \MyFont ...
```

In a similar fashion you can set up Zapfino, given that you have the professional version of this font.

```
\definefontfeature
  [zapfino]
  [mode=node,script=latn,language=dflt,
   liga=yes,clig=yes,calt=yes]

\font\MyFont=ZapfinoExtraLTPro*zapfino at 24pt
```

Here we show the more low level way of associating features with fonts. We advice users to stick to the more verbose and indirect way.

4 Status

Keep in mind that MkIV functionality is experimental, may change and in many cases is hidden for the user. The previous section may indicate that only fonts are dealt with, but there is more. Once stabelized new and extended functionality will be discussed in dedicated manuals. Some is already revealed in `mk.pdf` and in articles.

- You can read from zip archives and protocols like http and ftp. The prefix for zip files is `zip://`.
- Color and similar features will be reimplemented on top of the already present attributes framework.
- As soon as control over paragraph building is present, we will enable specialized hyphenation, spell checking, inter–character spacing, etc. These features work already but don't yet interplay nicely with some TEX internals that we need more control over.
- Logging can be done in xml format, error and debug messages can be reported in html popups.
- Currently we don't ship the xml parsers, but this will happen soon. These are written in Lua and will provide some specialized xml path searching.
- Grid snapping, line numbering, parallel texts and similar trickery has been experimented with (I have lots of test files) but will be built in in due time. Using Lua and node list manipulations for this demands splitting core files into MkII and MkIV parts and that takes a while.
- New features like weighted vertical spacing are ‘work in progress’ and need to be wrapped in a high level interface.
- Verbatim is rewritten and only TEX and METAPOST highlighting are supported. Once the interface for this has stabelized, we expect users to come up with other pretty printing code.

- There is a documentation module for Lua code. It works ok, but may be rewritten using the new lpeg feature.
- Index sorting is already done in Lua, so TEXutil (functionality) is no longer needed. Eventually TEXexec will be gone too, that is, it will be a call to mtxrun with a TEXexec script component. This means that a few years along the road the dependency on Ruby will be gone as well.

5 Help

As usual help can be asked for on the mailing list. Testing is appreciated and we expect the usual input for enhancements. For instance, Arabic is already kind of supported, but for Chinese I need to adapt some of the MkII trickery.

From now on the so called minimal ConTEXt distributions will contain LuaTEX. Once we have a stable MkIV we can consider lean and mean LuaTEX mininal that installs and runs from a zip file. It is already possible to (auto)mount trees in zip files, but what we really need is a real minimal main tree.