Dednat6: an extensible (semi-)preprocessor for LuaLaTeX that understands diagrams in ASCII art

Eduardo Ochs
UFF, Brazil
eduardoochs@gmail.com
http://angg.twu.net/dednat6.html

(La)TeX treats lines starting with “`%`” as comments, and ignores them. This means that we can put anything we want in these “`%`” lines, even code to be processed by other programs besides TeX.

In this talk we will describe a “semi-preprocessor”, called `dednat6`, that makes blocks of lines starting with “`%L`” be executed as Lua code, treats blocks of lines starting with “`%:`” as 2D representations of derivation trees, and treats blocks of lines starting with “`%D`” as diagrams in which a 2D representation specifies where the nodes are to be placed and a stack-based language inspired by Forth is used to connect these nodes with arrows.

A predecessor of `dednat6`, called `dednat4`, was a preprocessor in the usual sense: running “`dednat4.lua foo.tex`” on a shell would convert the trees and diagrams in “`%:`” and “`%D`”-blocks in `foo.tex` to “`\def`”s that LaTeX can understand, and would put these “`\def`”s in a file `foo.dnt`; we had to put in `foo.tex` an “`\input "foo.dnt"`” that would load those definitions. `Dednat6` does something almost equivalent to that, but it uses LuaLaTeX to avoid the need for an external preprocessor and for an auxiliar “`.dnt`” file. Here is how; the workflow is unusual, so let’s see it in detail.

Put a line “`\directlua{dofile("loaddednat6.lua")}`” in a file `bar.tex`. When we run “`lualatex bar.tex`” that line loads the dednat6 library, initializes the global variable `tf` in the Lua interpreter with a `TexFile` object, and sets `tf.nline=1` to indicate that nothing in `bar.tex` has been processed with dednat6 yet. A (low-level) command like `\directlua{processlines(200, 300)}` in `bar.tex` would “process the lines 200 to 300 in `bar.tex` with dednat6”, which means to take all the blocks of “`%L`”-lines, “`%:`”-lines, and “`%D`”-lines between the lines 200 to 300 in `bar.tex`, run them in the adequate interpreters, and then send the resulting LaTeX code — usually “`\def`”s — to the latex interpreter. The high-level macro “`\pu`” runs “`\directlua(processuntil{tex.inputlineno})`”, that runs `processlines` on the lines between `tf.nline=1` and the line where the current “`\pu`” is, and advances `tf.nline` — i.e., it processes with dednat6 the lines in the current file between the previous “`\pu`” and the current one.

The strings “`%L`”, “`%:`”, and “`%D`” are called “heads” in dednat6, and it’s easy to add support for new heads; this can even be done in a “`%L`” block.

Note that with dednat4 all the “`\def`”s had to be loaded at once; in dednat6 idioms like “`{\pu ...}`”, “`$\pu ...$`”, and “`$$\pu ...$$`” can be used to make the “`\def`”s between the last “`\pu`” and the current one be local.