1. CCBASE. This package provides the base macros and Lua module for my LuaTEX setup.

2. The package files.

⟨ `*{ccbase.tex}` 2 ⟩ ≡
  ⟨ TEX package preamble 4 ⟩
  ⟨ TEX macros 6 ⟩
  ⟨ TEX package postamble 5 ⟩

3. ⟨ `*{ccbase.lua}` 3 ⟩ ≡

```
local exports = {}
```
  ⟨ Lua global variables 11 ⟩
  ⟨ Lua functions 12 ⟩
```
return exports
```

4. ⟨ TEX package preamble 4 ⟩ ≡
  ⟨ Include guards 7 ⟩
```
\input eplain
\directlua{ccbase = dofile(kpse.find_file("ccbase.lua"))}
\makeatletter
```
This code is used in section 2.

5. ⟨ TEX package postamble 5 ⟩ ≡
```
\makeatother
\endinput
```
This code is used in section 2.

6. ⟨ TEX macros 6 ⟩ ≡
```
\def\typeout{\immediate\write17}
```
See also sections 8 and 9.

This code is used in section 2.

7. Include guards. We define a command `\pragmaonce{id}` that expands to nothing the first time it is called, and to `\endinput` otherwise. This is useful to provide include guards to our package files. Since we want to include guard `CCBASE` itself, this is the first thing we define, and the first thing we use.

⟨ Include guards 7 ⟩ ≡

```
\def\pragmaonce#1{
  \csname pragmaonce#1\endcsname%
  \global\expandafter\let\csname pragmaonce#1\endcsname=\endinput
}
\pragmaonce{ccbase}
```

This code is used in section 4.

8. Catcodes and verbatim. The definitions below are copied from `tugboat.cmn`, including the documentation: The following allow for easier changes of category. These require that the character be addressed as a control-sequence: e.g. `\makeescape\/` will make the `/` an escape character.

⟨ TeX macros 6 ⟩ +≡

```
\def\makeescape#1{\catcode`#1=0 }
\def\makebgroup#1{\catcode`#1=1 }
\def\makeegroup#1{\catcode`#1=2 }
\def\makemath#1{\catcode`#1=3 }
\def\makealign#1{\catcode`#1=4 }
\def\makeeol#1{\catcode`#1=5 }
\def\makeparm#1{\catcode`#1=6 }
\def\makesup#1{\catcode`#1=7 }
\def\makesub#1{\catcode`#1=8 }
\def\makeignore#1{\catcode`#1=9 }
\def\makespace#1{\catcode`#1=10 }
\def\makeletter#1{\catcode`#1=11 }
\def\makeother#1{\catcode`#1=12 }
\def\makeactive#1{\catcode`#1=13 }
\def\makecomment#1{\catcode`#1=14 }
```

9. Inline Lua code. This comes directly from LuaTEX's "Writing Lua in TeX" page, using the catcode routines defined before for simplicity. It introduces two macros, `\luacode` and `\endluacode`, that are used as a begin-end environment.

To syntax highlight Lua code inside TeX, create a `$VIMFILES/after/syntax/plaintex.vim` file with the following contents:

```
unlet b:current_syntax
syn include @LUA syntax/lua.vim

syn region luatex matchgroup=contextIdentifier
    \ start='\\luacode'
    \ end='\\endluacode'
    \ contains=@LUA
```

The reason I define `\luacode` below with an `\expandafter` is to make Vim properly syntax-highlight `CCBASE` itself.

⟨ TEX macros 6 ⟩ +≡

```
\expandafter\def\csname luacode\endcsname{
  \bgroup
  \makeother\{
  \makeother\}
  \makeother\^^M
  \makeother\#
  \makeother\~
  \makeother\%
  \doluacode
}
\bgroup
\makeother\^^M %
\long\gdef\doluacode#1^^M#2\endluacode{\directlua{#2}\egroup}%
\egroup
```

10. The output routine.

11. LuaTeX nodes. TEX entities are represented in LuaTEX as nodes of different types. Here we define a few global variables that make type identification more efficient later on.

⟨ Lua global variables 11 ⟩ ≡

```
local GLUE_TYPE    = node.id("glue")
local GLYPH_TYPE   = node.id("glyph")
local HLIST_TYPE   = node.id("hlist")
local KERN_TYPE    = node.id("kern")
local MATH_TYPE    = node.id("math")
local RULE_TYPE    = node.id("rule")
local VLIST_TYPE   = node.id("vlist")
local WHATSIT_TYPE = node.id("whatsit")
```

See also sections 13 and 14.

This code is used in section 3.

12. ⟨ Lua functions 12 ⟩ ≡

See also sections 15, 16, and 17.

This code is used in section 3.

13. ⟨ Lua global variables 11 ⟩ +≡

```
exports["GLUE_TYPE"]    = GLUE_TYPE
exports["GLYPH_TYPE"]   = GLYPH_TYPE
exports["HLIST_TYPE"]   = HLIST_TYPE
exports["KERN_TYPE"]    = KERN_TYPE
exports["MATH_TYPE"]    = MATH_TYPE
exports["RULE_TYPE"]    = RULE_TYPE
exports["VLIST_TYPE"]   = VLIST_TYPE
exports["WHATSIT_TYPE"] = WHATSIT_TYPE
```

14. Dimensions.

⟨ Lua global variables 11 ⟩ +≡

```
local dims = {
  ["sp"] = 1,
  ["pt"] = 2^16,
  ["pc"] = 12*2^16,
  ["bp"] = 72*2^16,
  ["in"] = 72.27*2^16,
}
```

15. ⟨ Lua functions 12 ⟩ +≡

```
local function dim2str(value,from,to)
  return string.format("%f"..to,value*dims[from]/dims[to])
end
exports["dim2str"] = dim2str
```

16. ⟨ Lua functions 12 ⟩ +≡

```
local function str2dim(value,to)
  value = value:gsub("^[ \t]*","")
  value = value:gsub("[ \t].*$","")
  local from = value:gsub("[-0-9.]+","")
  value = value:gsub("[^-0-9.]+","")
  return tonumber(value)*dims[from]/dims[to]
end
exports["str2dim"] = str2dim
```

17. ⟨ Lua functions 12 ⟩ +≡

```
local function mkglue(w,st,sto,sh,sho)
  local glue = node.new(ccbase.GLUE_TYPE)
  glue.spec = node.new("glue_spec")
  glue.spec.width = w
  glue.spec.stretch = st
  glue.spec.stretch_order = sto
  glue.spec.shrink = sh
  glue.spec.shrink_order = sho
  return glue
end
exports["mkglue"] = mkglue
```

CCBASE