The luatex package

Heiko Oberdiek
<heiko.oberdiek at googlemail.com>

2010/03/09 v0.4

Abstract

This package manages the new and extended features and resources that LuaTeX provides. Examples are attributes and catcode tables.

Contents

1 Documentation

1.1 Introduction

TEX provides global resources such as registers. But it does not provide an interface for managing these resources. For example, two packages want to use a counter register. If they take the same register number, then the use of both packages will conflict and they cannot be used together. Therefore formats such as plain TEX or LATEX implement an allocation scheme for registers. A package reserves with `\newcount` an unused register number for its own exclusive use.

Nowadays TEX is not alone anymore: ε-TEX, pdfTEX and other compilers for TEX are developed that extend and add new features and resources.

Now LuaTEX has reached beta state. It inherits most of pdfTEX's features including ε-TEX. Also it implements new concepts such as attributes or catcode tables.

1.1.1 LATEX

LATEX 2$_\varepsilon$ is frozen and therefore refuses to even notice the new TEX variants. Not even the old ε-TEX is supported by its kernel. At least there is a third party package etex that manages the new ε-TEX resources.

This package tries to do the same for LuaTEX and starts to support at least a few of the new features.

1.1.2 plain TEX

LATEX has inherited its resource handling from plain TEX. The interface is basically the same: `\newcount`, . . . Therefore this package tries to follow this tradition by providing compatibility to plain TEX. It can be loaded with plain TEX and defines at least some of the features that this packages provides for LATEX.

1.2 Register allocation

1.2.1 Register with 16 bit

Because LuaTEX is a super set of ε-TEX regarding registers, the register allocation scheme should not conflict with package `etex`. Therefore this package is loaded to inherit its allocation scheme. The only change is currently that the limit is increased to 65536 registers for the following register classes:

- count
- dimen
- skip
- muskip
- marks
- toks
- box

This affects the number of global and local registers. Because it is done in a package and not in the kernel, it is possible that someone loads package `etex` before uses the local allocation variants. This will prevent the extension for this register class. If more registers are needed, just load package `luatex` earlier.

1.2.2 Insertions

Insertions need four registers `\count`, `\dimen`, `\skip`, and `\box` with the same number. Usually they are allocated downwards from 254, 253, ... Also `\newcount`, `\newdimen`, ... fill up these register numbers from below before switching to higher register numbers by package `etex`. When this occurs, no insertions can be allocated anymore.

Therefore `\newcount`, `\newdimen`, `\newskip`, and `\newbox` are replaced by their global variants (`\globcount`, ...) that use the higher numbers immediately, leaving the room for insertions. There should not be an efficiency penalty because LuaTEX stores the registers of a class in the same Lua table unlike ε-TEX, where registers below 256 are stored in an array and higher numbers are put in a tree structure.

1.3 Attributes

Nodes can have custom attributes in LuaTEX. These attributes are organized by a new register class. As the other registers up to 2^{16} attributes are supported. An attribute value can be negative that means the attribute is not set. Otherwise TEX's range of non-negative integers up to 2^31 are available.

`\newattribute` {⟨*cmd*⟩}

Macro `\newattribute` defines command ⟨*cmd*⟩ using `\attributedef` using an new attribute number. The new attribute is initially unset.

`\setattribute` {⟨*cmd*⟩} {⟨*value*⟩}

Macro `\setattribute` locally sets attribute command ⟨*cmd*⟩ to the number ⟨*value*⟩. Valid values range from -1 until 2^31 (the upper limit is the same as for other TEX integer numbers).

`\unsetattribute` {⟨*cmd*⟩}

Macro `\unsettattribute` clears the attribute command ⟨*cmd*⟩.

1.4 Catcode tables

LuaTeX introduces catcode tables as new feature, see documentation. There is need for discussion, how to deal best:

- `\initcatcodetable` and `\setcatcodetable` act globally.
- `\catcodetable` causes an error if used with an uninitialized catcode table.
- Large catcode table numbers should be avoided because of performance breakdown.
- Use case LaTeX package: The package must not be surprised by changed catcodes and must not surprise by changing catcodes accidently. Catcode tables could offer a solution. At the begin a catcode regime with standard catcodes is established and the old one is restored afterwards.
- Use case: LuaTeX's `tex.print` might be used with a catcode table number, for example a table where all entries have catcode "other".
- Readonly catcode tables.
- Is there is a need for local allocations? (Package `etex`'s `\loc` variants are not used in TeX Live 2007.)

1.4.1 Interface proposal

The idea: `\newcatcodetable` allocates odd numbered catcode tables. Even numbered tables are managed as stack. Also some catcode tables are defined. These must not be changed.

```
\newcatcodetable {⟨cmd⟩}
```

Macro `\newcatcodetable` reserves a new catcode table and remembers its number in ⟨*cmd*⟩. The catcode table is initialized with ini-TeX's catcodes.

```
\CatcodeTableIniTeX
\CatcodeTableString
\CatcodeTableOther
\CatcodeTableLaTeX
```

These are catcode tables and must not be changed. `\CatcodeTableIniTeX` contains the catcode settings of ini-TeX. `\CatcodeTableString` follows TeX's convention of `\string`, `\meaning` and friends. The space gets catcode 10 (space), the other characters have catcode 12 (other). In `\CatcodeTableOther` all entries have catcode 12 (other). `\CatcodeTableLaTeX` contains the setting of a pure LaTeX format ('at' is other).

```
\CatcodeTableStack
\IncCatcodeTableStack
\DecCatcodeTableStack
```

`\CatcodeTableStack` is the stack pointer. Initially it is catcode table zero. `\IncCatcodeTableStack` and `\DecCatcodeTableStack` increments and decrements the stack pointer. Currently `\IncCatcodeTableStack` does not initialize a new catcode table. Both increment and decrement operations do not set a catcode table.

```
\PushCatcodeTableNumStack
\PopCatcodeTableNumStack
```

It can be handy to have a global stack for catcode table numbers to deal with the global assignment property of `\initcatcodetable` and `\savecatcodetable`. `\PushCatcodeTableNumStack` pushes the current catcode table on the stack. `\PopCatcodeTableNumStack` pops the topmost number off the number stack to set the current catcode table. Catcode table zero is used in case of an empty stack.

```
\BeginCatcodeRegime {⟨catcodetable⟩}
\EndCatcodeRegime
```

`\BeginCatcodeRegime` remembers the current catcode table number. Then it creates and uses a fresh catcode table on the stack that is initialized by ⟨*catcodetable*⟩:

```
\PushCatcodeTableNumStack
\catcodetable⟨catcodetable⟩ \IncCatcodeTableStack
\savecatcodetable\CatcodeTableStack
\catcodetable\CatcodeTableStack
```

`\EndCatcodeRegime` drops the catcode table, created by `\BeginCatcodeRegime` and sets the catcode table that was active before:

```
\DecCatcodeTableStack
\PopCatcodeTableNumStack
```

These macros solve the use case, described earlier for a LaTeX package:

```
% package foobar.sty
\BeginCatcodeRegime\CatcodeTableLaTeX
\makeatletter
% ... package contents ...
\EndCatcodeRegime
% end of package
```

If the package wants to change catcodes after its loading, `\AtBeginDocument` or `\AtEndOfPackage` can be used.

```
\SetCatcodeRange {⟨from⟩} {⟨to⟩} {⟨catcode⟩}
```

The catcodes of characters in range from ⟨*from*⟩ to inclusive ⟨*to*⟩ are set to ⟨*catcode*⟩.

1.5 Lua module loading

Currently LuaTeX (version 0.20) does not support Lua script files inside `TDS:scripts//`, because Lua's mechanism for module loading does not use the kpathsea library. Therefore this packages appends a kpse loader to the list of Lua's module loaders. It finds the module ⟨*module*⟩ by

```
kpse.find_file("⟨module⟩.lua", "texmfscripts")
```

Unhappily kpathsea does not support directory components in a file name. Therefore the Lua convention is not followed to replace dots in the module name by the directory separator.

Example: A Lua script of a package foobar wants the following modules:

```
require("foobar.hello.world")
require("org.somewhere.xyz")
```

Then they can be find in:

```
TDS:scripts/foobar/foobar.hello.world.lua
TDS:scripts/foobar/org.somewhere.xyz.lua
```

I would have preferred the following locations, following lua conventions, e. g.:

```
TDS:scripts/foobar/hello/world.lua
TDS:scripts/foobar/org/somewhere/xyz.lua
```

But I do not know, how to achieve this in a reliable way using kpathsea.

1.5.1 Package luatex-loader

If someone do not need or want package luatex but it's extension for module loading, then he can use package luatex-loader. Both plain TEX and LATEX are supported.

2 Implementation

```
⟨*package⟩
```

2.1 Reload check and package identification

Reload check, especially if the package is not used with LATEX.

```
\begingroup
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\let\expandafter\x\csname ver@luatex.sty\endcsname
  \ifx\x\relax % plain-TeX, first loading
  \else
    \def\empty{}%
    \ifx\x\empty % LaTeX, first loading,
      % variable is initialized, but \ProvidesPackage not yet seen
    \else
      \catcode35 6 % #
      \expandafter\ifx\csname PackageInfo\endcsname\relax
        \def\x#1#2{%
          \immediate\write-1{Package #1 Info: #2.}%
        }%
      \else
        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
      \fi
      \x{luatex}{The package is already loaded}%
      \aftergroup\endinput
    \fi
  \fi
\endgroup
```

Package identification:

```
\begingroup
  \catcode35 6 % #
  \catcode40 12 % (
  \catcode41 12 % )
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode47 12 % /
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode91 12 % [
  \catcode93 12 % ]
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\@undefined
        \xdef#1{#3}%
      \fi
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@luatex.sty\endcsname
\ProvidesPackage{luatex}%
  [2010/03/09 v0.4 LuaTeX basic definition package (HO)]
```

2.2 Catcodes

```
\begingroup
  \catcode123 1 % {
  \catcode125 2 % }
  \def\x{\endgroup
    \expandafter\edef\csname LuT@AtEnd\endcsname{%
      \catcode35 \the\catcode35\relax
      \catcode64 \the\catcode64\relax
      \catcode123 \the\catcode123\relax
      \catcode125 \the\catcode125\relax
    }%
  }%
\x
\catcode35 6 % #
\catcode64 11 % @
\catcode123 1 % {
\catcode125 2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\LuT@AtEnd{%
    \LuT@AtEnd
    \catcode#1 \the\catcode#1\relax
  }%
  \catcode#1 #2\relax
}
\TMP@EnsureCode{10}{12}% ^^J
\TMP@EnsureCode{34}{12}% "
\TMP@EnsureCode{36}{3}% $
\TMP@EnsureCode{39}{12}% ’
\TMP@EnsureCode{40}{12}% (
\TMP@EnsureCode{41}{12}% )
\TMP@EnsureCode{42}{12}% *
\TMP@EnsureCode{43}{12}% +
\TMP@EnsureCode{44}{12}% ,
\TMP@EnsureCode{45}{12}% -
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{60}{12}% <
```

```
\TMP@EnsureCode{61}{12}% =
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{95}{12}% _ (other!)
\TMP@EnsureCode{96}{12}% ‘
```

2.3 Check for LuaTEX

Without LuaTEX there is no point in using this package.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input infwarerr.sty\relax
  \input ifluatex.sty\relax
\else
  \RequirePackage{infwarerr}[2007/09/09]%
  \RequirePackage{ifluatex}[2009/04/10]%
\fi
```

```
\ifluatex
\else
  \@PackageError{luatex}{%
    This package may only be run using LuaTeX%
  }\@ehc
  \LuT@AtEnd
  \expandafter\endinput
\fi
```

2.4 Provide LuaTEX primitives

```
\ifnum\luatexversion<36 %
  \def\LuT@MakePrimitive#1{%
    \expandafter\let\csname luatex#1\expandafter\endcsname
    \csname #1\endcsname
  }%
\else
  \def\LuT@MakeLuatexPrimitive#1{%
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname luatex#1\endcsname\relax
      \begingroup\expandafter\expandafter\expandafter\endgroup
      \expandafter\ifx\csname #1\endcsname\relax
      \else
        \expandafter\let
        \csname luatex#1\expandafter\endcsname
        \csname #1\endcsname
      \fi
    \fi
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname luatex#1\endcsname\relax
      \begingroup
        \expandafter\let\csname luatex#1\endcsname\@undefined
        \ifnum0%
          \directlua{%
            if tex.enableprimitives then %
              tex.enableprimitives(’luatex’,{’#1’})%
              tex.print(’1’)%
            end%
          }%
          \expandafter\ifx\csname luatex#1\endcsname\relax\else1\fi
        =11 %
          \global\expandafter\let
          \csname luatex#1\expandafter\endcsname
          \csname luatex#1\endcsname
        \else
          \@PackageError{luatex}{%
      tex.enableprimitives failed for ‘#1’%
     }\@ehc
    \fi
   \endgroup
  \fi
 }%
 \def\LuT@MakePrimitive#1{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname#1\endcsname\relax
   \begingroup
    \expandafter\let\csname#1\endcsname\@undefined
    \ifnum0%
     \directlua{%
      if tex.enableprimitives then %
       tex.enableprimitives('',{'#1'})%
       tex.print('1')%
      end%
     }%
     \expandafter\ifx\csname#1\endcsname\relax\else1\fi
    =11 %
     \global\expandafter\let
     \csname#1\expandafter\endcsname
     \csname#1\endcsname
    \else
     \@PackageError{luatex}{%
      tex.enableprimitives failed for ‘#1’%
     }\@ehc
    \fi
   \endgroup
  \fi
 }%
\fi
\LuT@MakeLuatexPrimitive{attribute}
\LuT@MakeLuatexPrimitive{attributedef}
\LuT@MakeLuatexPrimitive{catcodetable}
\LuT@MakeLuatexPrimitive{initcatcodetable}
\LuT@MakeLuatexPrimitive{luaescapestring}
\LuT@MakeLuatexPrimitive{savecatcodetable}
\LuT@MakePrimitive{numexpr}
```

2.5 Inherit support for ε-TEX

Package etex is not compatible for plain TEX. But it could be present if a format is used that is based on `etex.src`. Therefore we only load the package in case of LATEX and tests its presence independently of the format by looking for `\et@xins`.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
\else
 \RequirePackage{etex}[1998/03/26]%
\fi
```

2.6 Adaption of ε-TEX's register allocation

ε-TEXhas increased the number of TEX registers from 2^8 (256) to 2^{15} (32768) for a register class. LuaTEX extends the limit further to $2^1$6 (65536). The allocation scheme of package etex is not changed. But this can be subject for discussion.

If a register class hasn't registered any local registers yet, then the limit can safely be pushed to 65536.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname et@xins\endcsname\relax
 \@PackageWarningNoLine{luatex}{%
  Support for eTeX is not loaded (etex.src)%
 }%
\else
 \def\LuT@temp#1{%
  \ifnum\count27#1=32768 %
    \count27#1=65536 %
  \fi
 }%
 \LuT@temp0%
 \LuT@temp1%
 \LuT@temp2%
 \LuT@temp3%
 \LuT@temp4%
 \LuT@temp5%
 \LuT@temp6%
```

ε-TeX uses an array for the first 256 registers and then a tree structure. LuaTeX stores all registers of a class in one Lua table. There shouldn't be large performance differences. This allows starting immediately in the extended area, leaving room for insertions.

```
  \let\newcount\globcount
  \let\newdimen\globdimen
  \let\newskip\globskip
  \let\newbox\globbox
\fi
```

2.7 plain TeX compatibility

\@empty

```
\expandafter\ifx\csname @empty\endcsname\relax
  \def\@empty{}%
\fi
```

\@gobble

```
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
```

\@firstofone

```
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
```

\@firstoftwo

```
\expandafter\ifx\csname @firstoftwo\endcsname\relax
  \long\def\@firstoftwo#1#2{#1}%
\fi
```

\@car

```
\expandafter\ifx\csname @car\endcsname\relax
  \def\@car#1#2\@nil{#1}%
\fi
```

\@cdr

```
\expandafter\ifx\csname @cdr\endcsname\relax
  \def\@cdr#1#2\@nil{#2}%
\fi
```

\@ifstar

```
\expandafter\ifx\csname @ifstar\endcsname\relax
  \def\@ifstar#1{%
    \@ifnextchar*{\@firstoftwo{#1}}%
  }%
```

\@ifnextchar

```
  \long\def\@ifnextchar#1#2#3{%
    \let\reserved@d=#1%
    \def\reserved@a{#2}%
    \def\reserved@b{#3}%
    \futurelet\@let@token\@ifnch
  }%
```

\@ifnch

```
  \def\@ifnch{%
    \ifx\@let@token\@sptoken
      \let\reserved@c\@xifnch
    \else
      \ifx\@let@token\reserved@d
        \let\reserved@c\reserved@a
      \else
        \let\reserved@c\reserved@b
      \fi
    \fi
    \reserved@c
  }%
```

\@sptoken

```
  \let\LuT@temp\:%
  \def\:{\let\@sptoken= }%
  \: % explicit space
```

\@xifnch

```
  \def\:{\@xifnch}%
  \expandafter\def\: {%
    \futurelet\@let@token\@ifnch
  }%
  \let\:\LuT@temp
\fi
```

\@tempcnta

```
\expandafter\ifx\csname @tempcnta\endcsname\relax
  \csname newcount\endcsname\@tempcnta
\fi
```

\@tempcntb

```
\expandafter\ifx\csname @tempcntb\endcsname\relax
  \csname newcount\endcsname\@tempcntb
\fi
```

\LuT@newcommand

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname newcommand\endcsname\relax
  \def\LuT@newcommand#1[#2]#3{%
    \ifx#1\@undefined
      \let#1\relax
    \else
      \ifx#1\relax
      \else
        \@PackageError{luatex}{%
          \string#1 is already defined.\MessageBreak
          Redefinition is skipped%
        }\@ehc
      \fi
    \fi
    \ifx#1\relax
```

```
      \ifcase#2 %
        \def#1{#3}%
      \or
        \def#1##1{#3}%
      \or
        \def#1##1##2{#3}%
      \or
        \def#1##1##2##3{#3}%
      \or
        \@INTERNAL@ERROR
      \fi
    \fi
  }%
\else
  \def\LuT@newcommand{\newcommand*}%
\fi
```

2.8 Attributes

2.8.1 Allocation

\LuT@AllocAttribute

```
\newcount\LuT@AllocAttribute
\LuT@AllocAttribute=\m@ne
```

\newattribute

```
\LuT@newcommand\newattribute[1]{%
  \ifnum\LuT@AllocAttribute<65535 %
    \global\advance\LuT@AllocAttribute\@ne
    \allocationnumber\LuT@AllocAttribute
    \global\luatexattributedef#1=\allocationnumber
    \unsetattribute{#1}%
    \wlog{\string#1=\string\attribute\the\allocationnumber}%
  \else
    \errmessage{No room for a new \string\attribute}%
  \fi
}
```

2.8.2 Interface

\setattribute

```
\LuT@newcommand\setattribute[2]{%
  #1=\numexpr#2\relax
}
```

\unsetattribute

```
\ifnum\luatexversion<37
  \LuT@newcommand\LuT@UnsetAttributeValue[0]{}%
  \let\LuT@UnsetAttributeValue\m@ne
\else
  \LuT@newcommand\LuT@UnsetAttributeValue[0]{-2147483647 }%
\fi
\LuT@newcommand\unsetattribute[1]{%
  #1=\LuT@UnsetAttributeValue
}
```

2.9 Catcode tables

2.9.1 Allocation

\LuT@AllocCatcodeTable

```
\newcount\LuT@AllocCatcodeTable
\LuT@AllocCatcodeTable=\m@ne
\newcount\CatcodeTableStack
\CatcodeTableStack=\z@
```

\newcatcodetable

```
\LuT@newcommand\newcatcodetable[1]{%
  \ifnum\LuT@AllocCatcodeTable<1114110 % 0x10FFFF is maximal \chardef
    % or < 268435455 % 2^28 - 1
    \global\advance\LuT@AllocCatcodeTable by\tw@
    \allocationnumber=\LuT@AllocCatcodeTable
    \global\chardef#1=\allocationnumber
    \wlog{%
      \string#1=\string\catcodetable\the\allocationnumber
    }%
  \else
    \errmessage{No room for a new \string\catcodetable}%
  \fi
}%
```

\IncCatcodeTableStack

```
\LuT@newcommand\IncCatcodeTableStack[0]{%
  \ifnum\CatcodeTableStack<268435454 %
    \global\advance\CatcodeTableStack by\tw@
  \else
    \@PackageError{luatex}{%
      Catcode table stack overflow%
    }\@ehd
  \fi
}
```

\DecCatcodeTableStack

```
\LuT@newcommand\DecCatcodeTableStack[0]{%
  \ifnum\CatcodeTableStack>\z@
    \global\advance\CatcodeTableStack by-2 %
  \else
    \@PackageError{luatex}{%
      Catcode table stack is empty%
    }\@ehd
  \fi
}
```

2.9.2 \SetCatcodeRange

\SetCatcodeRange

```
\LuT@newcommand\SetCatcodeRange[3]{%
  \edef\LuT@temp{%
    \noexpand\@tempcnta=\the\@tempcnta
    \noexpand\@tempcntb=\the\@tempcntb
    \noexpand\count@=\the\count@
    \relax
  }%
  \@tempcnta=\numexpr#1\relax
  \@tempcntb=\numexpr#2\relax
  \count@=\numexpr#3\relax
  \loop
    \unless\ifnum\@tempcnta>\@tempcntb
    \catcode\@tempcnta=\count@
    \advance\@tempcnta by \@ne
  \repeat
  \LuT@temp
}
```

2.9.3 Predefined catcode tables

```
\newcatcodetable\CatcodeTableIniTeX
\newcatcodetable\CatcodeTableString
\newcatcodetable\CatcodeTableOther
\newcatcodetable\CatcodeTableLaTeX
```

```
\luatexinitcatcodetable\CatcodeTableIniTeX
\begingroup
  \def\@makeother#1{\catcode#1=12\relax}%
  \@firstofone{%
    \luatexcatcodetable\CatcodeTableIniTeX
    \begingroup
      \SetCatcodeRange{0}{8}{15}%
      \catcode9=10 % tab
      \catcode11=15 %
      \catcode12=13 % form feed
      \SetCatcodeRange{14}{31}{15}%
      \catcode35=6 % hash
      \catcode36=3 % dollar
      \catcode38=4 % ampersand
      \catcode94=7 % circumflex
      \catcode95=8 % underscore
      \catcode123=1 % brace left
      \catcode125=2 % brace right
      \catcode126=13 % tilde
      \catcode127=15 %
      \luatexsavecatcodetable\CatcodeTableLaTeX
    \endgroup
    \@makeother{0}% nul
    \@makeother{13}% carriage return
    \@makeother{37}% percent
    \@makeother{92}% backslash
    \@makeother{127}%
    \SetCatcodeRange{65}{90}{12}% A-Z
    \SetCatcodeRange{97}{122}{12}% a-z
    \luatexsavecatcodetable\CatcodeTableString
    \@makeother{32}% space
    \luatexsavecatcodetable\CatcodeTableOther
  \endgroup
}%
```

2.9.4 Number stack

`\LuT@NumStackEmpty` A special empty stack value because of `\@cdr`'s brace removal.

```
\def\LuT@NumStackEmpty{0}
```

`\LuT@NumStack`

```
\let\LuT@NumStack\LuT@NumStackEmpty
```

`\PushCatcodeTableNumStack`

```
\LuT@newcommand\PushCatcodeTableNumStack[0]{%
  \xdef\LuT@NumStack{%
    {\the\luatexcatcodetable}\LuT@NumStack
  }%
}
```

`\PopCatcodeTableNumStack`

```
\LuT@newcommand\PopCatcodeTableNumStack[0]{%
  \ifx\LuT@NumStack\LuT@NumStackEmpty
    \@PackageWarning{luatex}{Empty catcode table number stack}%
    \luatexcatcodetable\z@
  \else
```

```
    \luatexcatcodetable=\expandafter\@car\LuT@NumStack\@nil\relax
    \xdef\LuT@NumStack{%
      \expandafter\@cdr\LuT@NumStack\@nil
    }%
  \fi
}
```

2.9.5 Catcode regime macros

\BeginCatcodeRegime

```
\LuT@newcommand\BeginCatcodeRegime[1]{%
  \PushCatcodeTableNumStack
  \luatexcatcodetable=\numexpr#1\relax
  \IncCatcodeTableStack
  \luatexsavecatcodetable\CatcodeTableStack
  \luatexcatcodetable\CatcodeTableStack
}
```

\EndCatcodeRegime

```
\LuT@newcommand\EndCatcodeRegime[0]{%
  \DecCatcodeTableStack
  \PopCatcodeTableNumStack
}
```

2.10 Lua module loader

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input luatex-loader.sty\relax
\else
  \RequirePackage{luatex-loader}[2010/03/09]%
\fi
```

```
\LuT@AtEnd
⟨/package⟩
```

```
⟨*loader⟩
```

Reload check, especially if the package is not used with LaTeX.

```
\begingroup
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\let\expandafter\x\csname ver@luatex-loader.sty\endcsname
  \ifx\x\relax % plain-TeX, first loading
  \else
    \def\empty{}%
    \ifx\x\empty % LaTeX, first loading,
      % variable is initialized, but \ProvidesPackage not yet seen
    \else
      \catcode35 6 % #
      \expandafter\ifx\csname PackageInfo\endcsname\relax
        \def\x#1#2{%
          \immediate\write-1{Package #1 Info: #2.}%
        }%
      \else
        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
      \fi
      \x{luatex-loader}{The package is already loaded}%
      \aftergroup\endinput
    \fi
  \fi
\endgroup
```

Package identification:

```
\begingroup
  \catcode35 6 % #
  \catcode40 12 % (
  \catcode41 12 % )
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode47 12 % /
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode91 12 % [
  \catcode93 12 % ]
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\@undefined
        \xdef#1{#3}%
      \fi
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@luatex-loader.sty\endcsname
\ProvidesPackage{luatex-loader}%
  [2010/03/09 v0.4 Lua module loader (HO)]
```

```
\begingroup
  \catcode10 12 % ^^J
  \catcode34 12 % "
  \catcode39 12 % '
  \catcode40 12 % (
  \catcode41 12 % )
  \catcode44 12 % ,
  \catcode46 12 % .
  \catcode60 12 % <
  \catcode61 12 % =
  \catcode95 12 % _ (other!)
  \catcode96 12 % `
  \endlinechar=10 %
  \ifnum\luatexversion<36 %
    \directlua0%
  \else %
    \expandafter\directlua %
  \fi %
  {%
     do
      local script = "oberdiek.luatex.lua"
      local file = kpse.find_file(script, "texmfscripts")
      if file then
        texio.write_nl("(" .. file .. ")")
        dofile(file)
      else
        error("File `" .. script .. "' not found")
      end
    end
  }%
\endgroup%
```

⟨/loader⟩

2.11 Lua script

Currently LuaTEX does not use KPSE when searching for module files. The following Lua script implements a workaround. It extends `package.loader` by another search method. Modules are found by the module name with extension `.lua` similar to

```
kpsewhich --format=texmfscripts ⟨module⟩.lua
```

Unhappily kpsewhich does not support directory components in the file name. Therefore a module `a.b.c` cannot be installed as `a/b/c.lua`. The script must be named `a.b.c.lua`.

⟨*lua⟩

```
module("oberdiek.luatex", package.seeall)
```

```
function kpse_module_loader(module)
  local script = module .. ".lua"
  local file = kpse.find_file(script, "texmfscripts")
  if file then
    local loader, error = loadfile(file)
    if loader then
      texio.write_nl("(" .. file .. ")")
      return loader
    end
    return "\n\t[oberdiek.luatex.kpse_module_loader] Loading error:\n\t"
           .. error
  end
  return "\n\t[oberdiek.luatex.kpse_module_loader] Search failed"
end
table.insert(package.loaders, kpse_module_loader)
```

⟨/lua⟩

3 Test

⟨*test2⟩

```
\documentclass{article}
\def\LoadCommand{%
  \RequirePackage{luatex}[2010/03/09]%
}
```

⟨/test2⟩

⟨*test3⟩

```
\documentclass{article}
\def\LoadCommand{%
  \RequirePackage{luatex-loader}[2010/03/09]%
}
```

⟨/test3⟩

3.1 Catcode checks for loading

⟨*test1⟩

```
\catcode`\{=1 %
\catcode`\}=2 %
```

```
\catcode‘\#=6 %
\catcode‘\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
  \countdef\count@=255 %
\fi
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
\expandafter\ifx\csname loop\endcsname\relax
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
{%
  \def\loop#1\repeat{%
    \def\body{#1}%
    \iterate
  }%
  \def\iterate{%
    \body
      \let\next\iterate
    \else
      \let\next\relax
    \fi
    \next
  }%
  \let\repeat=\fi
}%
\def\RestoreCatcodes{}
\count@=0 %
\loop
  \edef\RestoreCatcodes{%
    \RestoreCatcodes
    \catcode\the\count@=\the\catcode\count@\relax
  }%
\ifnum\count@<255 %
  \advance\count@ 1 %
\repeat

\def\RangeCatcodeInvalid#1#2{%
  \count@=#1\relax
  \loop
    \catcode\count@=15 %
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input luatex.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode‘\@=12 %
  \catcode‘\\=0 %
  \catcode‘\{=1 %
  \catcode‘\}=2 %
  \catcode`\#=6 %
  \catcode`\[=12 %
  \catcode`\]=12 %
  \catcode`\%=14 %
  \catcode`\ =10 %
  \catcode13=5 %
  \LoadCommand
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```

⟨/test1⟩

3.2 Catcode tables

3.2.1 Predefined catcode tables

⟨*test4⟩

```
\NeedsTeXFormat{LaTeX2e}
```

Remember LaTeX's initial catcodes in count registers starting at `\TestLaTeX`.

```
\count0=0 %
\chardef\TestLaTeX=1000 %
\chardef\TestMax=300 %
\loop
  \count\numexpr\TestLaTeX+\count0\relax=\catcode\count0 %
\ifnum\count0<\TestMax
  \advance\count0 by 1 %
\repeat
\documentclass{minimal}
\usepackage{luatex}[2010/03/09]
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}
\makeatletter
\def\Check#1{%
  \Expect*{\the\count@=\the\catcode\count@}%
         *{\the\count@=#1}%
}
\newcount\scratch
\def\Test#1#2{%
  \begin{qstest}{CatcodeTable#1}{CatcodeTable#1}%
    \luatexcatcodetable\csname CatcodeTable#1\endcsname
    \count@=\z@
    \loop
      \scratch=#2\relax
      \Expect*{\the\count@=\the\catcode\count@}%
             *{\the\count@=\the\scratch}%
    \ifnum\count@<\TestMax
       \advance\count@\@ne
    \repeat
  \end{qstest}%
}
\Test{LaTeX}{\the\count\numexpr\TestLaTeX+\count@}
\Test{String}{\ifnum\count@=32 10\else 12\fi}
\Test{Other}{12}
\luatexinitcatcodetable99 %
\Test{IniTeX}{%
  0\relax
  \begingroup
    \luatexcatcodetable99 %
    \global\scratch=\the\catcode\count@
  \endgroup
}
```

3.2.2 Catcode table number stack

```
\begin{qstest}{CatcodeTableNumStack}{CatcodeTableNumStack}
  \def\TestStack#1{%
    \Expect*{\LuT@NumStack}{#1}%
  }%
  \TestStack{0}%
  \PushCatcodeTableNumStack
  \TestStack{{0}0}%
  \@firstofone{%
    \begingroup
      \luatexinitcatcodetable12 %
      \luatexcatcodetable12 %
      \PushCatcodeTableNumStack
      \TestStack{{12}{0}0}%
      \PopCatcodeTableNumStack
      \TestStack{{0}0}%
      \PopCatcodeTableNumStack
      \TestStack{0}%
      \def\TestWarning{Missing empty stack warning}%
      \def\@PackageWarning#1#2{\def\TestWarning{empty stack}}%
      \PopCatcodeTableNumStack
      \TestStack{0}%
      \Expect*{\TestWarning}{empty stack}%
    \endgroup
  }%
\end{qstest}
```

3.2.3 Catcode table stack

```
\begin{qstest}{CatcodeTableStack}{CatcodeTableStack}
  \def\TestStack#1{%
    \Expect*{\the\CatcodeTableStack}{#1}%
  }%
  \TestStack{0}%
  \IncCatcodeTableStack
  \TestStack{2}%
  \IncCatcodeTableStack
  \TestStack{4}%
  \begingroup
    \IncCatcodeTableStack
    \TestStack{6}%
  \endgroup
  \TestStack{6}%
  \begingroup
    \DecCatcodeTableStack
    \TestStack{4}%
  \endgroup
  \TestStack{4}%
  \DecCatcodeTableStack
  \TestStack{2}%
  \DecCatcodeTableStack
  \TestStack{0}%
  \begingroup
    \def\TestError{Missing error}%
    \def\@PackageError#1#2#3{%
      \def\TestError{Empty stack}%
    }%
    \DecCatcodeTableStack
    \TestStack{0}%
    \Expect*{\TestError}{Empty stack}%
  \endgroup
\end{qstest}
```

3.2.4 Catcode regime macros

```
\begin{qstest}{CatcodeRegime}{CatcodeRegime}
  \def\TestStacks#1#2#3{%
    \Expect*{\the\luatexcatcodetable}{#1}%
    \Expect*{\the\CatcodeTableStack}{#2}%
    \Expect*{\LuT@NumStack}{#3}%
  }%
  \TestStacks{0}{0}{0}%
  \catcode`\|=7 %
  \BeginCatcodeRegime\CatcodeTableLaTeX
    \TestStacks{2}{2}{{0}0}%
    \Expect*{\the\catcode`\|}{12}%
  \EndCatcodeRegime
  \TestStacks{0}{0}{0}%
  \Expect*{\the\catcode`\|}{7}%
\end{qstest}
```

3.3 Attribute allocation

```
\begin{qstest}{Attributes}{Attributes}
  \newattribute\TestAttr
  \Expect*{\meaning\TestAttr}%
         *{\string\attribute\number\allocationnumber}%
  \Expect*{\the\allocationnumber}{0}%
  \begingroup
    \newattribute\TestAttr
    \Expect*{\the\allocationnumber}{1}%
  \endgroup
  \Expect*{\the\allocationnumber}{0}%
  \Expect*{\meaning\TestAttr}*{\string\attribute1}%
  \Expect*{\the\TestAttr}*{\number\LuT@UnsetAttributeValue}%
  \def\Test#1{%
    \setattribute\TestAttr{#1}%
    \Expect*{\the\TestAttr}{#1}%
  }%
  \Test{0}%
  \Test{1}%
  \Test{-1}%
  \Test{123}%
  \unsetattribute\TestAttr
  \Expect*{\the\TestAttr}*{\number\LuT@UnsetAttributeValue}%
  \begingroup
    \Expect*{\the\TestAttr}*{\number\LuT@UnsetAttributeValue}%
    \Test{1234}%
  \endgroup
  \Expect*{\the\TestAttr}*{\number\LuT@UnsetAttributeValue}%
\end{qstest}
```

```
\@@end
```

⟨/test4⟩

3.4 Short test for plain TEX

⟨*test5⟩

```
\input luatex.sty\relax
\newattribute\TestAttr
\setattribute\TestAttr{10}
\unsetattribute\TestAttr
\newcatcodetable\TestCTa
\begingroup
  \SetCatcodeRange{`A}{`Z}{12}%
\endgroup
\BeginCatcodeRegime\CatcodeTableLaTeX
\EndCatcodeRegime
```

```
\end
⟨/test5⟩
```

4 Installation

4.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/luatex.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/luatex.pdf Documentation.

Bundle. All the packages of the bundle 'oberdiek' are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

CTAN:install/macros/latex/contrib/oberdiek.tds.zip

TDS refers to the standard "A Directory Structure for TEX Files" (CTAN:tds/tds.pdf). Directories with `texmf` in their name are usually organized this way.

4.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package attachfile2 comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting docstrip archive. The files are extracted by running the `.dtx` through plain TEX:

```
tex luatex.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

luatex.sty	→	tex/generic/oberdiek/luatex.sty
luatex-loader.sty	→	tex/generic/oberdiek/luatex-loader.sty
oberdiek.luatex.lua	→	scripts/oberdiek/oberdiek.luatex.lua
luatex.pdf	→	doc/latex/oberdiek/luatex.pdf
test/luatex-test1.tex	→	doc/latex/oberdiek/test/luatex-test1.tex
test/luatex-test2.tex	→	doc/latex/oberdiek/test/luatex-test2.tex
test/luatex-test3.tex	→	doc/latex/oberdiek/test/luatex-test3.tex
test/luatex-test4.tex	→	doc/latex/oberdiek/test/luatex-test4.tex
test/luatex-test5.tex	→	doc/latex/oberdiek/test/luatex-test5.tex
luatex.dtx	→	source/latex/oberdiek/luatex.dtx

If you have a `docstrip.cfg` that configures and enables docstrip's TDS installing feature, then some files can already be in the right place, see the documentation of docstrip.

[1] ftp://ftp.ctan.org/tex-archive/

4.4 Refresh file name databases

If your TeX distribution (teTeX, mikTeX, …) relies on file name databases, you must refresh these. For example, teTeX users run `texhash` or `mktexlsr`.

4.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is pdftk, e.g. unpack the file into the current directory:

```
pdftk luatex.pdf unpack_files output .
```

Unpacking with LaTeX. The `.dtx` chooses its action depending on the format:

plain TeX: Run docstrip and extract the files.

LaTeX: Generate the documentation.

If you insist on using LaTeX for docstrip (really, docstrip does not need LaTeX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{luatex.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfLaTeX:

```
pdflatex luatex.dtx
makeindex -s gind.ist luatex.idx
pdflatex luatex.dtx
makeindex -s gind.ist luatex.idx
pdflatex luatex.dtx
```

5 History

[2007/12/12 v0.1]

- First public version.

[2009/04/10 v0.2]

- Requires package ifluatex in version 2.0 to ensure `\luatexversion`.
- Updates the call of `\directlua`, the syntax has changed in LuaTeX 0.36.

[2009/12/02 v0.3]

- Unsetting of attributes updated for LuaTeX 0.37.

[2010/03/09 v0.4]

- Support for lua states removed.
- Calling tex.enableprimitives for used primitives.

6 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.