The pdftexcmds package

Heiko Oberdiek*

2026-10-28 v0.34

Abstract

LuaTeX provides most of the commands of pdfTeX 1.40. However a number of utility functions are removed. This package tries to fill the gap and implements some of the missing primitives using Lua.

Contents

1 Documentation **2**
- 1.1 General principles . . . 3
- 1.2 Macros . . . 3
  - 1.2.1 Strings . . . 3
  - 1.2.2 Files . . . 4
  - 1.2.3 Timekeeping . . . 4
  - 1.2.4 Miscellaneous . . . 5
  - 1.2.5 Additional macro: `\pdf@isprimitive` . . . 6
  - 1.2.6 Experimental . . . 6

2 Implementation **6**
- 2.1 Reload check and package identification . . . 7
- 2.2 Catcodes . . . 8
- 2.3 Load packages . . . 9
- 2.4 Without LuaTeX . . . 9
- 2.5 `\pdf@primitive`, `\pdf@ifprimitive` . . . 11
  - 2.5.1 Using LuaTeX's `tex.enableprimitives` . . . 11
  - 2.5.2 Trying various names to find the primitives . . . 11
  - 2.5.3 Result . . . 12
- 2.6 XeTeX . . . 12
- 2.7 `\pdf@isprimitive` . . . 13
- 2.8 `\pdf@draftmode` . . . 14
- 2.9 Load Lua module . . . 15
- 2.10 Lua functions . . . 16
  - 2.10.1 Helper macros . . . 16
  - 2.10.2 Strings . . . 17
  - 2.10.3 Files . . . 18
  - 2.10.4 Timekeeping . . . 19
  - 2.10.5 Shell escape . . . 20
- 2.11 Lua module . . . 20
  - 2.11.1 Strings . . . 21
  - 2.11.2 Files . . . 23
  - 2.11.3 Timekeeping . . . 25
  - 2.11.4 Miscellaneous . . . 25

*Please report any issues at https://github.com/LaTeX-Package-Repositories/pdftexcmds/issues

3 Installation **26**
3.1 Download . . . 26
3.2 Package installation . . . 26
3.3 Refresh file name databases . . . 27
3.4 Some details for the interested . . . 27

4 References **27**

5 History **28**
[2007/11/11 v0.1] . . . 28
[2007/11/12 v0.2] . . . 28
[2007/12/12 v0.3] . . . 28
[2009/04/10 v0.4] . . . 28
[2009/09/22 v0.5] . . . 28
[2009/09/23 v0.6] . . . 28
[2009/12/12 v0.7] . . . 28
[2010/03/01 v0.8] . . . 28
[2010/04/01 v0.9] . . . 28
[2010/11/04 v0.10] . . . 28
[2010/11/11 v0.11] . . . 28
[2011/01/30 v0.12] . . . 28
[2011/03/04 v0.13] . . . 28
[2011/04/10 v0.14] . . . 29
[2011/04/16 v0.15] . . . 29
[2011/04/22 v0.16] . . . 29
[2011/06/29 v0.17] . . . 29
[2011/07/01 v0.18] . . . 29
[2011/07/28 v0.19] . . . 29
[2011/11/29 v0.20] . . . 29
[2016/05/10 v0.21] . . . 29
[2016/05/21 v0.22] . . . 29
[2016/10/02 v0.23] . . . 29
[2017/01/29 v0.24] . . . 29
[2017/03/19 v0.25] . . . 29
[2018/01/21 v0.26] . . . 29
[2018/01/30 v0.27] . . . 30
[2018/09/07 v0.28] . . . 30
[2018/09/10 v0.29] . . . 30
[2019/07/25 v0.30] . . . 30
[2019/11/24 v0.31] . . . 30
[2020-06-04 v0.32] . . . 30
[2020-06-24 v0.33] . . . 30
[2026-10-28 v0.34] . . . 30

6 Index **30**

1 Documentation

Some primitives of pdfTEX [1] are not defined by LuaTEX [2]. This package implements macro based solutions using Lua code for the following missing pdfTEX primitives;

- `\pdfstrcmp`
- `\pdfunescapehex`
- `\pdfescapehex`
- `\pdfescapename`
- `\pdfescapestring`
- `\pdffilesize`

- `\pdffilemoddate`
- `\pdffiledump`
- `\pdfmdfivesum`
- `\pdfresettimer`
- `\pdfelapsedtime`
- `\immediate\write18`

The original names of the primitives cannot be used:

- The syntax for their arguments cannot be easily simulated by macros. The primitives use key words such as `file` (`\pdfmdfivesum`) or `offset` and `length` (`\pdffiledump`) and use ⟨*general text*⟩ for the other arguments. Using token register assignments, ⟨*general text*⟩ could be handled. However, the simulated primitives are expandable and register assignments would destroy this important property. (⟨*general text*⟩ allows something like `\expandafter\bgroup ...}`.)
- The original primitives can be expanded using one expansion step. The new macros need two expansion steps because of the additional macro expansion. Example:

  ```
  \expandafter\foo\pdffilemoddate{file}
  ```
  vs.
  ```
  \expandafter\expandafter\expandafter
  \foo\pdf@filemoddate{file}
  ```

1.1 General principles

Naming convention: Usually this package defines a macro `\pdf@`⟨*cmd*⟩ if pdfTEX provides `\pdf`⟨*cmd*⟩.

Arguments: The order of arguments in `\pdf@`⟨*cmd*⟩ is the same as for the corresponding primitive of pdfTEX. The arguments are ordinary undelimited TEX arguments, no ⟨*general text*⟩ and without additional keywords.

Expandibility: The macro `\pdf@`⟨*cmd*⟩ is expandable if the corresponding pdfTEX primitive has this property. Exactly two expansion steps are necessary (first is the macro expansion) except for `\pdf@primitive` and `\pdf@ifprimitive`. The latter ones are not macros, but have the direct meaning of the primitive.

Without LuaTEX: The macros `\pdf@`⟨*cmd*⟩ are mapped to the commands of pdfTEX if they are available. Otherwise they are undefined.

Availability: The macros that the packages provides are undefined, if the necessary primitives are not found and cannot be implemented by Lua.

1.2 Macros

1.2.1 Strings [1, "7.15 Strings"]

`\pdf@strcmp` {⟨*stringA*⟩} {⟨*stringB*⟩}

Same as `\pdfstrcmp{`⟨*stringA*⟩`}{`⟨*stringB*⟩`}`.

`\pdf@unescapehex` {⟨*string*⟩}

Same as `\pdfunescapehex{`⟨*string*⟩`}`. The argument is a byte string given in hexadecimal notation. The result are character tokens from 0 until 255 with catcode 12 and the space with catcode 10.

`\pdf@escapehex` {⟨*string*⟩}
`\pdf@escapestring` {⟨*string*⟩}
`\pdf@escapename` {⟨*string*⟩}

Same as the primitives of pdfTEX. However pdfTEX does not know about characters with codes 256 and larger. Thus the string is treated as byte string, characters with more than eight bits are ignored.

1.2.2 Files [1, "7.18 Files"]

`\pdf@filesize` {⟨*filename*⟩}

Same as `\pdffilesize{`⟨*filename*⟩`}`.

`\pdf@filemoddate` {⟨*filename*⟩}

Same as `\pdffilemoddate{`⟨*filename*⟩`}`.

`\pdf@filedump` {⟨*offset*⟩} {⟨*length*⟩} {⟨*filename*⟩}

Same as `\pdffiledump offset` ⟨*offset*⟩ `length` ⟨*length*⟩ `{`⟨*filename*⟩`}`. Both ⟨*offset*⟩ and ⟨*length*⟩ must not be empty, but must be a valid TEX number.

`\pdf@mdfivesum` {⟨*string*⟩}

Same as `\pdfmdfivesum{`⟨*string*⟩`}`. Keyword `file` is supported by macro `\pdf@filemdfivesum`.

`\pdf@filemdfivesum` {⟨*filename*⟩}

Same as `\pdfmdfivesum file{`⟨*filename*⟩`}`.

1.2.3 Timekeeping [1, "7.17 Timekeeping"]

The timekeeping macros are based on Andy Thomas' work [3].

`\pdf@resettimer`

Same as `\pdfresettimer`, it resets the internal timer.

`\pdf@elapsedtime`

Same as `\pdfelapsedtime`. It behaves like a read-only integer. For printing purposes it can be prefixed by `\the` or `\number`. It measures the time in scaled seconds (seconds multiplied with 65536) since the latest call of `\pdf@resettimer` or start of program/package. The resolution, the shortest time interval that can be measured, depends on the program and system.

- pdfTEX with `gettimeofday`: $\geq 1/65536\,\mathrm{s}$
- pdfTEX with `ftime`: $\geq 1\,\mathrm{ms}$
- pdfTEX with `time`: $\geq 1\,\mathrm{s}$
- LuaTEX: $\geq 10\,\mathrm{ms}$
  (`os.clock()` returns a float number with two decimal digits in LuaTEX beta-0.70.1-2011061416 (rev 4277)).

1.2.4 Miscellaneous [1, "7.21 Miscellaneous"]

`\pdf@draftmode`

If the TEX compiler knows `\pdfdraftmode` or `\draftmode` (pdfTEX, LuaTEX), then `\pdf@draftmode` returns, whether this mode is enabled. The result is an implicit number: one means the draft mode is available and enabled. If the value is zero, then the mode is not active or `\pdfdraftmode` is not available. An explicit number is yielded by `\number\pdf@draftmode`. The macro cannot be used to change the mode, see `\pdf@setdraftmode`.

`\pdf@ifdraftmode` {⟨*true*⟩} {⟨*false*⟩}

If `\pdfdraftmode` is available and enabled, ⟨*true*⟩ is called, otherwise ⟨*false*⟩ is executed.

`\pdf@setdraftmode` {⟨*value*⟩}

Macro `\pdf@setdraftmode` expects the number zero or one as ⟨*value*⟩. Zero deactivates the mode and one enables the draft mode. The macro does not have an effect, if the feature `\pdfdraftmode` is not available.

`\pdf@shellescape`

Same as `\pdfshellescape`. It is or expands to `1` if external commands can be executed and `0` otherwise. In pdfTEX external commands must be enabled first by command line option or configuration option. In LuaTEX option `--safer` disables the execution of external commands.

In LuaTEX before 0.68.0 `\pdf@shellescape` is not available due to a bug in `os.execute()`. The argumentless form crashes in some circumstances with segmentation fault. (It is fixed in version 0.68.0 or revision 4167 of LuaTEX. and packported to some version of 0.67.0).

Hints for usage:

- Before its use `\pdf@shellescape` should be tested, whether it is available. Example with package ltxcmds (loaded by package pdftexcmds):

```
\ltx@IfUndefined{pdf@shellescape}{%
  % \pdf@shellescape is undefined
}{%
  % \pdf@shellescape is available
}
```

  Use `\ltx@ifundefined` in expandable contexts.

- `\pdf@shellescape` might be a numerical constant, expands to the primitive, or expands to a plain number. Therefore use it in contexts where these differences does not matter.

- Use in comparisons, e.g.:

```
\ifnum\pdf@shellescape=0 ...
```

- Print the number: `\number\pdf@shellescape`

`\pdf@system` {⟨*cmdline*⟩}

It is a wrapper for `\immediate\write18` in pdfTEX or `os.execute` in LuaTEX.

In theory `os.execute` returns a status number. But its meaning is quite undefined. Are there some reliable properties? Does it make sense to provide an user interface to this status exit code?

`\pdf@primitive` *\cmd*

Same as `\pdfprimitive` in pdfTeX or LuaTeX. In XeTeX the primitive is called `\primitive`. Despite the current definition of the command *\cmd*, it's meaning as primitive is used.

`\pdf@ifprimitive` *\cmd*

Same as `\ifpdfprimitive` in pdfTeX or LuaTeX. XeTeX calls it `\ifprimitive`. It is a switch that checks if the command *\cmd* has it's primitive meaning.

1.2.5 Additional macro: `\pdf@isprimitive`

`\pdf@isprimitive` *\cmd1 \cmd2* {⟨*true*⟩} {⟨*false*⟩}

If *\cmd1* has the primitive meaning given by the primitive name of *\cmd2*, then the argument ⟨*true*⟩ is executed, otherwise ⟨*false*⟩. The macro `\pdf@isprimitive` is expandable. Internally it checks the result of `\meaning` and is therefore available for all TeX variants, even the original TeX. Example with LaTeX:

```
\makeatletter
\pdf@isprimitive{@@input}{input}{%
  \typeout{\string\@@input\space is original\string\input}%
}{%
  \typeout{Oops, \string\@@input\space is not the %
           original\string\input}%
}
```

1.2.6 Experimental

`\pdf@unescapehexnative` {⟨*string*⟩}
`\pdf@escapehexnative` {⟨*string*⟩}
`\pdf@escapenamenative` {⟨*string*⟩}
`\pdf@mdfivesumnative` {⟨*string*⟩}

The variants without `native` in the macro name are supposed to be compatible with pdfTeX. However characters with more than eight bits are not supported and are ignored. If LuaTeX is running, then its UTF-8 coded strings are used. Thus the full unicode character range is supported. However the result differs from pdfTeX for characters with eight or more bits.

`\pdf@pipe` {⟨*cmdline*⟩}

It calls ⟨*cmdline*⟩ and returns the output of the external program in the usual manner as byte string (catcode 12, space with catcode 10). The Lua documentation says, that the used `io.popen` may not be available on all platforms. Then macro `\pdf@pipe` is undefined.

2 Implementation

1 ⟨*package⟩

2.1 Reload check and package identification

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

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode35=6 % #
  \catcode39=12 % '
  \catcode44=12 % ,
  \catcode45=12 % -
  \catcode46=12 % .
  \catcode58=12 % :
  \catcode64=11 % @
  \catcode123=1 % {
  \catcode125=2 % }
  \expandafter\let\expandafter\x\csname ver@pdftexcmds.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
      \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{pdftexcmds}{The package is already loaded}%
      \aftergroup\endinput
    \fi
  \fi
\endgroup%
```

Package identification:

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode35=6 % #
  \catcode39=12 % '
  \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@pdftexcmds.sty\endcsname
\ProvidesPackage{pdftexcmds}%
  [2026-10-28 v0.34 Utility functions of pdfTeX for LuaTeX (HO)]%
```

2.2 Catcodes

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode123=1 % {
  \catcode125=2 % }
  \catcode64=11 % @
  \def\x{\endgroup
    \expandafter\edef\csname pdftexcmds@AtEnd\endcsname{%
      \endlinechar=\the\endlinechar\relax
      \catcode13=\the\catcode13\relax
      \catcode32=\the\catcode32\relax
      \catcode35=\the\catcode35\relax
      \catcode61=\the\catcode61\relax
      \catcode64=\the\catcode64\relax
      \catcode123=\the\catcode123\relax
      \catcode125=\the\catcode125\relax
    }%
  }%
\x\catcode61\catcode48\catcode32=10\relax%
\catcode13=5 % ^^M
\endlinechar=13 %
\catcode35=6 % #
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\pdftexcmds@AtEnd{%
    \pdftexcmds@AtEnd
    \catcode#1=\the\catcode#1\relax
  }%
  \catcode#1=#2\relax
}
\TMP@EnsureCode{0}{12}%
\TMP@EnsureCode{1}{12}%
\TMP@EnsureCode{2}{12}%
\TMP@EnsureCode{10}{12}% ^^J
\TMP@EnsureCode{33}{12}% !
\TMP@EnsureCode{34}{12}% "
\TMP@EnsureCode{38}{4}% &
\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{58}{12}% :
\TMP@EnsureCode{60}{12}% <
```

```
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{91}{12}% [
\TMP@EnsureCode{93}{12}% ]
\TMP@EnsureCode{94}{7}% ^ (superscript)
\TMP@EnsureCode{95}{12}% _ (other)
\TMP@EnsureCode{96}{12}% `
\TMP@EnsureCode{126}{12}% ~ (other)
\edef\pdftexcmds@AtEnd{%
  \pdftexcmds@AtEnd
  \escapechar=\number\escapechar\relax
  \noexpand\endinput
}
\escapechar=92 %
```

2.3 Load packages

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \def\TMP@RequirePackage#1[#2]{%
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname ver@#1.sty\endcsname\relax
      \input #1.sty\relax
    \fi
  }%
  \TMP@RequirePackage{infwarerr}[2007/09/09]%
  \TMP@RequirePackage{iftex}[2019/11/07]%%
  \TMP@RequirePackage{ltxcmds}[2010/12/02]%
\else
  \RequirePackage{infwarerr}[2007/09/09]%
  \RequirePackage{iftex}[2019/11/07]%
  \RequirePackage{ltxcmds}[2010/12/02]%
\fi
```

2.4 Without LuaTeX

```
\ifluatex
   \ifcsname catcodetable@string\endcsname\else\input{ltluatex}\fi
\else
  \def\pdftexcmds@nopdftex{%
    \let\pdftexcmds@nopdftex\relax
  }%
  \def\pdftexcmds@temp#1{%
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname
         \expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname\relax
      \pdftexcmds@nopdftex
    \else
      \expandafter\def\csname pdf@#1\expandafter\endcsname
        \expandafter{%
        \csname\expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname
      }%
    \fi
  }%
  \pdftexcmds@temp{strcmp}%
  \pdftexcmds@temp{escapehex}%
  \let\pdf@escapehexnative\pdf@escapehex
  \pdftexcmds@temp{unescapehex}%
  \let\pdf@unescapehexnative\pdf@unescapehex
  \pdftexcmds@temp{escapestring}%
  \pdftexcmds@temp{escapename}%
  \pdftexcmds@temp{filesize}%
  \pdftexcmds@temp{filemoddate}%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname pdfshellescape\endcsname\relax
```

```
    \pdftexcmds@nopdftex
    \ltx@IfUndefined{pdftexversion}{%
    }{%
      \ifnum\pdftexversion>120 % 1.21a supports \ifeof18
        \ifeof18 %
          \chardef\pdf@shellescape=0 %
        \else
          \chardef\pdf@shellescape=1 %
        \fi
      \fi
    }%
  \else
    \def\pdf@shellescape{%
      \pdfshellescape
    }%
  \fi
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname pdffiledump\endcsname\relax
    \pdftexcmds@nopdftex
  \else
    \def\pdf@filedump#1#2#3{%
      \pdffiledump offset#1 length#2{#3}%
    }%
  \fi
  \ifxetex
    \def\pdf@filedump#1#2#3{%
      \filedump offset#1 length#2{#3}%
    }%
  \fi
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname mdfivesum\endcsname\relax
      \pdftexcmds@nopdftex
    \else
      \def\pdf@mdfivesum#{\mdfivesum}%
      \let\pdf@mdfivesumnative\pdf@mdfivesum
      \def\pdf@filemdfivesum#{\mdfivesum file}%
    \fi
  \else
    \def\pdf@mdfivesum#{\pdfmdfivesum}%
    \let\pdf@mdfivesumnative\pdf@mdfivesum
    \def\pdf@filemdfivesum#{\pdfmdfivesum file}%
  \fi
  \def\pdf@system#{%
    \immediate\write18%
  }%
  \def\pdftexcmds@temp#1{%
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname
        \expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname\relax
      \pdftexcmds@nopdftex
    \else
      \begingroup\expandafter\expandafter\expandafter\endgroup
      \expandafter\let\csname pdf@#1\expandafter\endcsname
      \csname\expandafter\ifx\csname pdf#1\endcsname\relax\else pdf\fi#1\endcsname
    \fi
  }%
  \pdftexcmds@temp{resettimer}%
  \pdftexcmds@temp{elapsedtime}%
\fi
```

2.5 \pdf@primitive, \pdf@ifprimitive

Since version 1.40.0 pdfTeX has `\pdfprimitive` and `\ifpdfprimitive`. And `\pdfprimitive` was fixed in version 1.40.4.

XeTeX provides them under the name `\primitive` and `\ifprimitive`. LuaTeX knows both name variants, but they have possibly to be enabled first (`tex.enableprimitives`).

Depending on the format TeX Live uses a prefix `luatex`.

Caution: `\let` must be used for the definition of the macros, especially because of `\ifpdfprimitive`.

2.5.1 Using LuaTeX's `tex.enableprimitives`

```
\ifluatex
\pdftexcmds@directlua
  \ifnum\luatexversion<36 %
    \def\pdftexcmds@directlua{\directlua0 }%
  \else
    \let\pdftexcmds@directlua\directlua
  \fi
  \begingroup
    \newlinechar=10 %
    \endlinechar=\newlinechar
    \pdftexcmds@directlua{%
      if tex.enableprimitives then
        tex.enableprimitives(
          'pdf@',
          {'primitive', 'ifprimitive', 'pdfdraftmode','draftmode'}
        )
        tex.enableprimitives('', {'luaescapestring'})
      end
    }%
  \endgroup %
\fi
```

2.5.2 Trying various names to find the primitives

```
\pdftexcmds@strip@prefix
\def\pdftexcmds@strip@prefix#1>{}
\def\pdftexcmds@temp#1#2#3{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname pdf@#1\endcsname\relax
    \begingroup
      \def\x{#3}%
      \edef\x{\expandafter\pdftexcmds@strip@prefix\meaning\x}%
      \escapechar=-1 %
      \edef\y{\expandafter\meaning\csname#2\endcsname}%
    \expandafter\endgroup
    \ifx\x\y
      \expandafter\let\csname pdf@#1\expandafter\endcsname
      \csname #2\endcsname
    \fi
  \fi
}
\pdf@primitive
\pdftexcmds@temp{primitive}{pdfprimitive}{pdfprimitive}% pdfTeX, oldLuaTeX
\pdftexcmds@temp{primitive}{primitive}{primitive}% XeTeX, luatex
\pdftexcmds@temp{primitive}{luatexprimitive}{pdfprimitive}% oldLuaTeX
\pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% oldLuaTeX
```

\pdf@ifprimitive

```
\pdftexcmds@temp{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, oldLuaTeX
\pdftexcmds@temp{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX, luatex
\pdftexcmds@temp{ifprimitive}{luatexifprimitive}{ifpdfprimitive}% oldLuaTeX
\pdftexcmds@temp{ifprimitive}{luatexifpdfprimitive}{ifpdfprimitive}% oldLuaTeX
```

Disable broken \pdfprimitive.

```
\ifluatex\else
\begingroup
  \expandafter\ifx\csname pdf@primitive\endcsname\relax
  \else
    \expandafter\ifx\csname pdftexversion\endcsname\relax
    \else
      \ifnum\pdftexversion=140 %
        \expandafter\ifx\csname pdftexrevision\endcsname\relax
        \else
          \ifnum\pdftexrevision<4 %
            \endgroup
            \let\pdf@primitive\@undefined
            \@PackageInfoNoLine{pdftexcmds}{%
              \string\pdf@primitive\space disabled, %
              because\MessageBreak
              \string\pdfprimitive\space is broken until pdfTeX 1.40.4%
            }%
            \begingroup
          \fi
        \fi
      \fi
    \fi
  \fi
\endgroup
\fi
```

2.5.3 Result

```
\begingroup
  \@PackageInfoNoLine{pdftexcmds}{%
    \string\pdf@primitive\space is %
    \expandafter\ifx\csname pdf@primitive\endcsname\relax not \fi
    available%
  }%
  \@PackageInfoNoLine{pdftexcmds}{%
    \string\pdf@ifprimitive\space is %
    \expandafter\ifx\csname pdf@ifprimitive\endcsname\relax not \fi
    available%
  }%
\endgroup
```

2.6 XeTeX

Look for primitives \shellescape, \strcmp.

```
\def\pdftexcmds@temp#1{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname pdf@#1\endcsname\relax
    \begingroup
      \escapechar=-1 %
      \edef\x{\expandafter\meaning\csname#1\endcsname}%
      \def\y{#1}%
      \def\z##1->{}%
      \edef\y{\expandafter\z\meaning\y}%
    \expandafter\endgroup
    \ifx\x\y
      \expandafter\def\csname pdf@#1\expandafter\endcsname
```

```
      \expandafter{%
        \csname#1\endcsname
      }%
    \fi
  \fi
}%
\pdftexcmds@temp{shellescape}%
\pdftexcmds@temp{strcmp}%
```

2.7 \pdf@isprimitive

```
\def\pdf@isprimitive{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname pdf@strcmp\endcsname\relax
    \long\def\pdf@isprimitive##1{%
      \expandafter\pdftexcmds@isprimitive\expandafter{\meaning##1}%
    }%
    \long\def\pdftexcmds@isprimitive##1##2{%
      \expandafter\pdftexcmds@@isprimitive\expandafter{\string##2}{##1}%
    }%
    \def\pdftexcmds@@isprimitive##1##2{%
      \ifnum0\pdftexcmds@equal##1\delimiter##2\delimiter=1 %
        \expandafter\ltx@firstoftwo
      \else
        \expandafter\ltx@secondoftwo
      \fi
    }%
    \def\pdftexcmds@equal##1##2\delimiter##3##4\delimiter{%
      \ifx##1##3%
        \ifx\relax##2##4\relax
          1%
        \else
          \ifx\relax##2\relax
          \else
            \ifx\relax##4\relax
            \else
              \pdftexcmds@equalcont{##2}{##4}%
            \fi
          \fi
        \fi
      \fi
    }%
    \def\pdftexcmds@equalcont##1{%
      \def\pdftexcmds@equalcont####1####2##1##1##1##1{%
        ##1##1##1##1%
        \pdftexcmds@equal####1\delimiter####2\delimiter
      }%
    }%
    \expandafter\pdftexcmds@equalcont\csname fi\endcsname
  \else
    \long\def\pdf@isprimitive##1##2{%
      \ifnum\pdf@strcmp{\meaning##1}{\string##2}=0 %
        \expandafter\ltx@firstoftwo
      \else
        \expandafter\ltx@secondoftwo
      \fi
    }%
  \fi
}
\ifluatex
\ifx\pdfdraftmode\@undefined
  \let\pdfdraftmode\draftmode
```

```
\fi
\else
  \pdf@isprimitive
\fi
```

2.8 \pdf@draftmode

```
\let\pdftexcmds@temp\ltx@zero %
\ltx@IfUndefined{pdfdraftmode}{%
  \@PackageInfoNoLine{pdftexcmds}{\ltx@backslashchar pdfdraftmode not found}%
}{%
  \ifpdf
    \let\pdftexcmds@temp\ltx@one
    \@PackageInfoNoLine{pdftexcmds}{\ltx@backslashchar pdfdraftmode found}%
  \else
    \@PackageInfoNoLine{pdftexcmds}{%
      \ltx@backslashchar pdfdraftmode is ignored in DVI mode%
    }%
  \fi
}
\ifcase\pdftexcmds@temp
```

`\pdf@draftmode`

```
  \let\pdf@draftmode\ltx@zero
```

`\pdf@ifdraftmode`

```
  \let\pdf@ifdraftmode\ltx@secondoftwo
```

`\pdftexcmds@setdraftmode`

```
  \def\pdftexcmds@setdraftmode#1{}%
\else
```

`\pdftexcmds@draftmode`

```
  \let\pdftexcmds@draftmode\pdfdraftmode
```

`\pdf@ifdraftmode`

```
  \def\pdf@ifdraftmode{%
    \ifnum\pdftexcmds@draftmode=\ltx@one
      \expandafter\ltx@firstoftwo
    \else
      \expandafter\ltx@secondoftwo
    \fi
  }%
```

`\pdf@draftmode`

```
  \def\pdf@draftmode{%
    \ifnum\pdftexcmds@draftmode=\ltx@one
      \expandafter\ltx@one
    \else
      \expandafter\ltx@zero
    \fi
  }%
```

`\pdftexcmds@setdraftmode`

```
  \def\pdftexcmds@setdraftmode#1{%
    \pdftexcmds@draftmode=#1\relax
  }%
\fi
```

\pdf@setdraftmode

```
\def\pdf@setdraftmode#1{%
  \begingroup
    \count\ltx@cclv=#1\relax
  \edef\x{\endgroup
    \noexpand\pdftexcmds@@setdraftmode{\the\count\ltx@cclv}%
  }%
  \x
}
```

\pdftexcmds@@setdraftmode

```
\def\pdftexcmds@@setdraftmode#1{%
  \ifcase#1 %
    \pdftexcmds@setdraftmode{#1}%
  \or
    \pdftexcmds@setdraftmode{#1}%
  \else
    \@PackageWarning{pdftexcmds}{%
      \string\pdf@setdraftmode: Ignoring\MessageBreak
      invalid value `#1'%
    }%
  \fi
}
```

2.9 Load Lua module

```
\ifluatex
\else
  \expandafter\pdftexcmds@AtEnd
\fi%
\pdftexcmds@directlua{%
  require("pdftexcmds")%
}
\ifnum\luatexversion>37 %
  \ifnum0%
      \pdftexcmds@directlua{%
        if status.ini_version then %
          tex.write("1")%
        end%
      }>0 %
    \everyjob\expandafter{%
      \the\everyjob
      \pdftexcmds@directlua{%
        require("pdftexcmds")%
      }%
    }%
  \fi
\fi
\begingroup
  \def\x{2026-10-28 v0.34}%
  \ltx@onelevel@sanitize\x
  \edef\y{%
    \pdftexcmds@directlua{%
      if oberdiek.pdftexcmds.getversion then %
        oberdiek.pdftexcmds.getversion()%
      end%
    }%
  }%
  \ifx\x\y
  \else
    \@PackageError{pdftexcmds}{%
      Wrong version of lua module.\MessageBreak
```

```
      Package version: \x\MessageBreak
      Lua module: \y
    }\@ehc
  \fi
\endgroup
```

2.10 Lua functions

2.10.1 Helper macros

\pdftexcmds@toks

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname newtoks\endcsname\relax
  \toksdef\pdftexcmds@toks=0 %
\else
  \csname newtoks\endcsname\pdftexcmds@toks
\fi
```

\pdftexcmds@Patch

```
\def\pdftexcmds@Patch{0}
\ifnum\luatexversion>40 %
  \ifnum\luatexversion<66 %
    \def\pdftexcmds@Patch{1}%
  \fi
\fi

\ifcase\pdftexcmds@Patch
  \catcode`\&=14 %
\else
  \catcode`\&=9 %
```

\pdftexcmds@PatchDecode

```
  \def\pdftexcmds@PatchDecode#1\@nil{%
    \pdftexcmds@DecodeA#1^^A^^A\@nil{}%
  }%
```

\pdftexcmds@DecodeA

```
  \def\pdftexcmds@DecodeA#1^^A^^A#2\@nil#3{%
    \ifx\relax#2\relax
      \ltx@ReturnAfterElseFi{%
        \pdftexcmds@DecodeB#3#1^^A^^B\@nil{}%
      }%
    \else
      \ltx@ReturnAfterFi{%
        \pdftexcmds@DecodeA#2\@nil{#3#1^^@}%
      }%
    \fi
  }%
```

\pdftexcmds@DecodeB

```
  \def\pdftexcmds@DecodeB#1^^A^^B#2\@nil#3{%
    \ifx\relax#2\relax%
      \ltx@ReturnAfterElseFi{%
        \ltx@zero
        #3#1%
      }%
    \else
      \ltx@ReturnAfterFi{%
        \pdftexcmds@DecodeB#2\@nil{#3#1^^A}%
      }%
    \fi
  }%

\fi
```

```
\ifnum\luatexversion<36 %
\else
  \catcode`\0=9 %
\fi
```

2.10.2 Strings [1, "7.15 Strings"]

\pdf@strcmp

```
\long\def\pdf@strcmp#1#2{%
  \directlua0{%
    oberdiek.pdftexcmds.strcmp("\luaescapestring{#1}",%
        "\luaescapestring{#2}")%
  }%
}%
\pdf@isprimitive
```

\pdf@escapehex

```
\long\def\pdf@escapehex#1{%
  \directlua0{%
    oberdiek.pdftexcmds.escapehex("\luaescapestring{#1}", "byte")%
  }%
}%
```

\pdf@escapehexnative

```
\long\def\pdf@escapehexnative#1{%
  \directlua0{%
    oberdiek.pdftexcmds.escapehex("\luaescapestring{#1}")%
  }%
}%
```

\pdf@unescapehex

```
\def\pdf@unescapehex#1{%
& \romannumeral\expandafter\pdftexcmds@PatchDecode
  \the\expandafter\pdftexcmds@toks
  \directlua0{%
    oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
    oberdiek.pdftexcmds.unescapehex("\luaescapestring{#1}", "byte", \pdftexcmds@Patch)%
  }%
& \@nil
}%
```

\pdf@unescapehexnative

```
\def\pdf@unescapehexnative#1{%
& \romannumeral\expandafter\pdftexcmds@PatchDecode
  \the\expandafter\pdftexcmds@toks
  \directlua0{%
    oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
    oberdiek.pdftexcmds.unescapehex("\luaescapestring{#1}", \pdftexcmds@Patch)%
  }%
& \@nil
}%
```

\pdf@escapestring

```
\long\def\pdf@escapestring#1{%
  \directlua0{%
    oberdiek.pdftexcmds.escapestring("\luaescapestring{#1}")%
  }%
}
```

\pdf@escapename

```
\long\def\pdf@escapename#1{%
  \directlua0{%
    oberdiek.pdftexcmds.escapename("\luaescapestring{#1}", "byte")%
  }%
}
```

\pdf@escapenamenative

```
\long\def\pdf@escapenamenative#1{%
  \directlua0{%
    oberdiek.pdftexcmds.escapename("\luaescapestring{#1}")%
  }%
}
```

2.10.3 Files [1, "7.18 Files"]

\pdf@filesize

```
\def\pdf@filesize#1{%
  \directlua0{%
    oberdiek.pdftexcmds.filesize("\luaescapestring{#1}")%
  }%
}
```

\pdf@filemoddate

```
\def\pdf@filemoddate#1{%
  \directlua0{%
    oberdiek.pdftexcmds.filemoddate("\luaescapestring{#1}")%
  }%
}
```

\pdf@filedump

```
\def\pdf@filedump#1#2#3{%
  \directlua0{%
    oberdiek.pdftexcmds.filedump("\luaescapestring{\number#1}",%
        "\luaescapestring{\number#2}",%
        "\luaescapestring{#3}")%
  }%
}%
```

\pdf@mdfivesum

```
\long\def\pdf@mdfivesum#1{%
  \directlua0{%
    oberdiek.pdftexcmds.mdfivesum("\luaescapestring{#1}", "byte")%
  }%
}%
```

\pdf@mdfivesumnative

```
\long\def\pdf@mdfivesumnative#1{%
  \directlua0{%
    oberdiek.pdftexcmds.mdfivesum("\luaescapestring{#1}")%
  }%
}%
```

\pdf@filemdfivesum

```
\def\pdf@filemdfivesum#1{%
  \directlua0{%
    oberdiek.pdftexcmds.filemdfivesum("\luaescapestring{#1}")%
  }%
}%
```

2.10.4 Timekeeping [1, "7.17 Timekeeping"]

\protected

```
\let\pdftexcmds@temp=Y%
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname protected\endcsname\relax
  \pdftexcmds@directlua0{%
    if tex.enableprimitives then %
      tex.enableprimitives('', {'protected'})%
    end%
  }%
\fi
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname protected\endcsname\relax
  \let\pdftexcmds@temp=N%
\fi
```

\numexpr

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname numexpr\endcsname\relax
  \pdftexcmds@directlua0{%
    if tex.enableprimitives then %
      tex.enableprimitives('', {'numexpr'})%
    end%
  }%
\fi
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname numexpr\endcsname\relax
  \let\pdftexcmds@temp=N%
\fi

\ifx\pdftexcmds@temp N%
  \@PackageWarningNoLine{pdftexcmds}{%
    Definitions of \ltx@backslashchar pdf@resettimer and%
    \MessageBreak
    \ltx@backslashchar pdf@elapsedtime are skipped, because%
    \MessageBreak
    e-TeX's \ltx@backslashchar protected or %
    \ltx@backslashchar numexpr are missing%
  }%
\else
```

\pdf@resettimer

```
  \protected\def\pdf@resettimer{%
    \pdftexcmds@directlua0{%
      oberdiek.pdftexcmds.resettimer()%
    }%
  }%
```

\pdf@elapsedtime

```
  \protected\def\pdf@elapsedtime{%
    \numexpr
      \pdftexcmds@directlua0{%
        oberdiek.pdftexcmds.elapsedtime()%
      }%
    \relax
  }%

\fi
```

2.10.5 Shell escape

\pdf@shellescape

```
\ifnum\luatexversion<68 %
\else
  \protected\edef\pdf@shellescape{%
   \numexpr\directlua{tex.sprint(%
         \number\catcodetable@string,status.shell_escape)}\relax}
\fi
```

\pdf@system

```
\def\pdf@system#1{%
  \directlua0{%
    oberdiek.pdftexcmds.system("\luaescapestring{#1}")%
  }%
}
```

\pdf@lastsystemstatus

```
\def\pdf@lastsystemstatus{%
  \directlua0{%
    oberdiek.pdftexcmds.lastsystemstatus()%
  }%
}
```

\pdf@lastsystemexit

```
\def\pdf@lastsystemexit{%
  \directlua0{%
    oberdiek.pdftexcmds.lastsystemexit()%
  }%
}

\catcode`\0=12 %
```

\pdf@pipe Check availability of `io.popen` first.

```
\ifnum0%
    \pdftexcmds@directlua{%
      if io.popen then %
        tex.write("1")%
      end%
    }%
    =1 %
  \def\pdf@pipe#1{%
&   \romannumeral\expandafter\pdftexcmds@PatchDecode
    \the\expandafter\pdftexcmds@toks
    \pdftexcmds@directlua{%
      oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
      oberdiek.pdftexcmds.pipe("\luaescapestring{#1}", \pdftexcmds@Patch)%
    }%
&   \@nil
  }%
\fi

\pdftexcmds@AtEnd%
```

⟨/package⟩

2.11 Lua module

⟨*lua⟩

```
oberdiek = oberdiek or {}
local pdftexcmds = oberdiek.pdftexcmds or {}
oberdiek.pdftexcmds = pdftexcmds
local systemexitstatus
```

```
function pdftexcmds.getversion()
  tex.write("2026-10-28 v0.34")
end
```

2.11.1 Strings [1, "7.15 Strings"]

```
function pdftexcmds.strcmp(A, B)
  if A == B then
    tex.write("0")
  elseif A < B then
    tex.write("-1")
  else
    tex.write("1")
  end
end
local function utf8_to_byte(str)
  local i = 0
  local n = string.len(str)
  local t = {}
  while i < n do
    i = i + 1
    local a = string.byte(str, i)
    if a < 128 then
      table.insert(t, string.char(a))
    else
      if a >= 192 and i < n then
        i = i + 1
        local b = string.byte(str, i)
        if b < 128 or b >= 192 then
          i = i - 1
        elseif a == 194 then
          table.insert(t, string.char(b))
        elseif a == 195 then
          table.insert(t, string.char(b + 64))
        end
      end
    end
  end
  return table.concat(t)
end
function pdftexcmds.escapehex(str, mode)
  if mode == "byte" then
    str = utf8_to_byte(str)
  end
  tex.write((string.gsub(str, ".",
    function (ch)
      return string.format("%02X", string.byte(ch))
    end
  )))
end
```

See procedure `unescapehex` in file `utils.c` of pdfTEX. Caution: `tex.write` ignores leading spaces.

```
function pdftexcmds.unescapehex(str, mode, patch)
  local a = 0
  local first = true
  local result = {}
  for i = 1, string.len(str), 1 do
    local ch = string.byte(str, i)
    if ch >= 48 and ch <= 57 then
      ch = ch - 48
    elseif ch >= 65 and ch <= 70 then
      ch = ch - 55
    elseif ch >= 97 and ch <= 102 then
```

```
      ch = ch - 87
    else
      ch = nil
    end
    if ch then
      if first then
        a = ch * 16
        first = false
      else
        table.insert(result, a + ch)
        first = true
      end
    end
  end
  if not first then
    table.insert(result, a)
  end
  if patch == 1 then
    local temp = {}
    for i, a in ipairs(result) do
      if a == 0 then
        table.insert(temp, 1)
        table.insert(temp, 1)
      else
        if a == 1 then
          table.insert(temp, 1)
          table.insert(temp, 2)
        else
          table.insert(temp, a)
        end
      end
    end
    result = temp
  end
  if mode == "byte" then
    local utf8 = {}
    for i, a in ipairs(result) do
      if a < 128 then
        table.insert(utf8, a)
      else
        if a < 192 then
          table.insert(utf8, 194)
          a = a - 128
        else
          table.insert(utf8, 195)
          a = a - 192
        end
        table.insert(utf8, a + 128)
      end
    end
    result = utf8
  end
```

this next line added for current luatex; this is the only change in the file. eroux, 28apr13. (v 0.21)

```
  local unpack = _G["unpack"] or table.unpack
  tex.settoks(pdftexcmds.toks, string.char(unpack(result)))
end
```

See procedure `escapestring` in file `utils.c` of pdfTEX.

```
function pdftexcmds.escapestring(str, mode)
  if mode == "byte" then
    str = utf8_to_byte(str)
  end
```

```
  tex.write((string.gsub(str, ".",
    function (ch)
      local b = string.byte(ch)
      if b < 33 or b > 126 then
        return string.format("\\%.3o", b)
      end
      if b == 40 or b == 41 or b == 92 then
        return "\\" .. ch
      end
```

Lua 5.1 returns the match in case of return value `nil`.

```
      return nil
    end
  )))
end
```

See procedure `escapename` in file `utils.c` of pdfTEX.

```
function pdftexcmds.escapename(str, mode)
  if mode == "byte" then
    str = utf8_to_byte(str)
  end
  tex.write((string.gsub(str, ".",
    function (ch)
      local b = string.byte(ch)
      if b == 0 then
```

In Lua 5.0 `nil` could be used for the empty string, But `nil` returns the match in Lua 5.1, thus we use the empty string explicitly.

```
        return ""
      end
      if b <= 32 or b >= 127
          or b == 35 or b == 37 or b == 40 or b == 41
          or b == 47 or b == 60 or b == 62 or b == 91
          or b == 93 or b == 123 or b == 125 then
        return string.format("#%.2X", b)
      else
```

Lua 5.1 returns the match in case of return value `nil`.

```
        return nil
      end
    end
  )))
end
```

2.11.2 Files [1, "7.18 Files"]

```
function pdftexcmds.filesize(filename)
  local foundfile = kpse.find_file(filename, "tex", true)
  if foundfile then
    local size = lfs.attributes(foundfile, "size")
    if size then
      tex.write(size)
    end
  end
end
```

See procedure `makepdftime` in file `utils.c` of pdfTEX.

```
function pdftexcmds.filemoddate(filename)
  local foundfile = kpse.find_file(filename, "tex", true)
  if foundfile then
    local date = lfs.attributes(foundfile, "modification")
    if date then
      local d = os.date("*t", date)
      if d.sec >= 60 then
        d.sec = 59
      end
      local u = os.date("!*t", date)
```

```
      local off = 60 * (d.hour - u.hour) + d.min - u.min
      if d.year ~= u.year then
        if d.year > u.year then
          off = off + 1440
        else
          off = off - 1440
        end
      elseif d.yday ~= u.yday then
        if d.yday > u.yday then
          off = off + 1440
        else
          off = off - 1440
        end
      end
      local timezone
      if off == 0 then
        timezone = "Z"
      else
        local hours = math.floor(off / 60)
        local mins = math.abs(off - hours * 60)
        timezone = string.format("%+03d'%02d'", hours, mins)
      end
      tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",
          d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
    end
  end
end
function pdftexcmds.filedump(offset, length, filename)
  length = tonumber(length)
  if length and length > 0 then
    local foundfile = kpse.find_file(filename, "tex", true)
    if foundfile then
      offset = tonumber(offset)
      if not offset then
        offset = 0
      end
      local filehandle = io.open(foundfile, "rb")
      if filehandle then
        if offset > 0 then
          filehandle:seek("set", offset)
        end
        local dump = filehandle:read(length)
        pdftexcmds.escapehex(dump)
        filehandle:close()
      end
    end
  end
end
function pdftexcmds.mdfivesum(str, mode)
  if mode == "byte" then
    str = utf8_to_byte(str)
  end
  pdftexcmds.escapehex(md5.sum(str))
end
function pdftexcmds.filemdfivesum(filename)
  local foundfile = kpse.find_file(filename, "tex", true)
  if foundfile then
    local filehandle = io.open(foundfile, "rb")
    if filehandle then
      local contents = filehandle:read("*a")
      pdftexcmds.escapehex(md5.sum(contents))
      filehandle:close()
```

```
    end
  end
end
```

2.11.3 Timekeeping [1, "7.17 Timekeeping"]

The functions for timekeeping are based on Andy Thomas' work [3]. Changes:

- Overflow check is added.
- `string.format` is used to avoid exponential number representation for sure.
- `tex.write` is used instead of `tex.print` to get tokens with catcode 12 and without appended `\endlinechar`.

```
local basetime = 0
function pdftexcmds.resettimer()
  basetime = os.clock()
end
function pdftexcmds.elapsedtime()
  local val = (os.clock() - basetime) * 65536 + .5
  if val > 2147483647 then
    val = 2147483647
  end
  tex.write(string.format("%d", math.floor(val)))
end
```

2.11.4 Miscellaneous [1, "7.21 Miscellaneous"]

```
function pdftexcmds.shellescape()
  if os.execute then
    if status
        and status.luatex_version
        and status.luatex_version >= 68 then
      tex.write(os.execute())
    else
      local result = os.execute()
      if result == 0 then
        tex.write("0")
      else
        if result == nil then
          tex.write("0")
        else
          tex.write("1")
        end
      end
    end
  else
    tex.write("0")
  end
end
function pdftexcmds.system(cmdline)
  systemexitstatus = nil
  texio.write_nl("log", "system(" .. cmdline .. ") ")
  if os.execute then
    texio.write("log", "executed.")
    systemexitstatus = os.execute(cmdline)
  else
    texio.write("log", "disabled.")
  end
end
function pdftexcmds.lastsystemstatus()
  local result = tonumber(systemexitstatus)
  if result then
    local x = math.floor(result / 256)
```

```
997     tex.write(result - 256 * math.floor(result / 256))
998   end
999 end
1000 function pdftexcmds.lastsystemexit()
1001   local result = tonumber(systemexitstatus)
1002   if result then
1003     tex.write(math.floor(result / 256))
1004   end
1005 end
1006 function pdftexcmds.pipe(cmdline, patch)
1007   local result
1008   systemexitstatus = nil
1009   texio.write_nl("log", "pipe(" .. cmdline ..") ")
1010   if io.popen then
1011     texio.write("log", "executed.")
1012     local handle = io.popen(cmdline, "r")
1013     if handle then
1014       result = handle:read("*a")
1015       handle:close()
1016     end
1017   else
1018     texio.write("log", "disabled.")
1019   end
1020   if result then
1021     if patch == 1 then
1022       local temp = {}
1023       for i, a in ipairs(result) do
1024         if a == 0 then
1025           table.insert(temp, 1)
1026           table.insert(temp, 1)
1027         else
1028           if a == 1 then
1029             table.insert(temp, 1)
1030             table.insert(temp, 2)
1031           else
1032             table.insert(temp, a)
1033           end
1034         end
1035       end
1036       result = temp
1037     end
1038     tex.settoks(pdftexcmds.toks, result)
1039   else
1040     tex.settoks(pdftexcmds.toks, "")
1041   end
1042 end
```

1043 ⟨/lua⟩

3 Installation

3.1 Download

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

CTAN:macros/latex/contrib/pdftexcmds/pdftexcmds.dtx The source file.

CTAN:macros/latex/contrib/pdftexcmds/pdftexcmds.pdf Documentation.

3.2 Package installation

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

[1] CTAN:pkg/pdftexcmds

```
tex pdftexcmds.dtx
```

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

```
pdftexcmds.sty → tex/generic/pdftexcmds/pdftexcmds.sty
pdftexcmds.lua → tex/generic/pdftexcmds/pdftexcmds.lua
pdftexcmds.pdf → doc/latex/pdftexcmds/pdftexcmds.pdf
pdftexcmds.dtx → source/latex/pdftexcmds/pdftexcmds.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.

3.3 Refresh file name databases

If your TEX distribution (TEX Live, MiKTEX, ...) relies on file name databases, you must refresh these. For example, TEX Live users run `texhash` or `mktexlsr`.

3.4 Some details for the interested

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{pdftexcmds.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 pdftexcmds.dtx
bibtex pdftexcmds.aux
makeindex -s gind.ist pdftexcmds.idx
pdflatex pdftexcmds.dtx
makeindex -s gind.ist pdftexcmds.idx
pdflatex pdftexcmds.dtx
```

4 References

[1] Hàn Thế Thành et al. *The pdfTEX user manual*. Version 655 (1.40.11). 2010-11-23. URL: http://mirror.ctan.org/systems/pdftex/manual/pdftex-a.pdf (visited on 2011-11-29).

[2] LuaTEX development team. *LuaTEX Reference*. Version beta 0.71.0. 2011-10-11. URL: http://www.luatex.org/svn/trunk/manual/luatex.pdf (visited on 2011-11-29).

[3] Andy Thomas. *Analog of \pdfelapsedtime for LuaTEX and XETEX*. URL: http://tex.stackexchange.com/a/32531 (visited on 2011-11-29).

5 History

[2007/11/11 v0.1]

- First version.

[2007/11/12 v0.2]

- Short description fixed.

[2007/12/12 v0.3]

- Organization of Lua code as module.

[2009/04/10 v0.4]

- Adaptation for syntax change of `\directlua` in LuaTeX 0.36.

[2009/09/22 v0.5]

- `\pdf@primitive`, `\pdf@ifprimitive` added.
- XeTeX's variants are detected for `\pdf@shellescape`, `\pdf@strcmp`, `\pdf@primitive`, `\pdf@ifprimitive`.

[2009/09/23 v0.6]

- Macro `\pdf@isprimitive` added.

[2009/12/12 v0.7]

- Short info shortened.

[2010/03/01 v0.8]

- Required date for package ifluatex updated.

[2010/04/01 v0.9]

- Use `\ifeof18` for defining `\pdf@shellescape` between pdfTeX 1.21a (inclusive) and 1.30.0 (exclusive).

[2010/11/04 v0.10]

- `\pdf@draftmode`, `\pdf@ifdraftmode` and `\pdf@setdraftmode` added.

[2010/11/11 v0.11]

- Missing `\RequirePackage` for package ifpdf added.

[2011/01/30 v0.12]

- Already loaded package files are not input in plain TeX.

[2011/03/04 v0.13]

- Improved Lua function `shellescape` that also uses the result of `os.execute()` (thanks to Philipp Stephani).

[2011/04/10 v0.14]

- Version check of loaded module added.
- Patch for bug in LuaTeX between 0.40.6 and 0.65 that is fixed in revision 4096.

[2011/04/16 v0.15]

- LuaTeX: `\pdf@shellescape` is only supported for version 0.70.0 and higher due to a bug, `os.execute()` crashes in some circumstances. Fixed in LuaTeX beta-0.70.0, revision 4167.

[2011/04/22 v0.16]

- Previous fix was not working due to a wrong catcode of digit zero (due to easily support the old `\directlua0`). The version border is lowered to 0.68, because some beta-0.67.0 seems also to work.

[2011/06/29 v0.17]

- Documentation addition to `\pdf@shellescape`.

[2011/07/01 v0.18]

- Add Lua module loading in `\everyjob` for iniTeX (LuaTeX only).

[2011/07/28 v0.19]

- Missing space in an info message added (Martin Münch).

[2011/11/29 v0.20]

- `\pdf@resettimer` and `\pdf@elapsedtime` added (thanks Andy Thomas).

[2016/05/10 v0.21]

- local unpack added (thanks Élie Roux).

[2016/05/21 v0.22]

- adjust `\textbackslash` usage in bib file for biber bug.

[2016/10/02 v0.23]

- add file.close to lua filehandles (github pull request).

[2017/01/29 v0.24]

- Avoid loading luatex-loader for current luatex. (Use pdftexcmds.lua not oberdiek.pdftexcmds.lua to simplify file search with standard require)

[2017/03/19 v0.25]

- New `\pdf@shellescape` for LuaTeX, see github issue 20.

[2018/01/21 v0.26]

- use rb not r mode for file open github issue 34.

[2018/01/30 v0.27]

- `\pdf@mdfivesum` for XeTeX

[2018/09/07 v0.28]

- Fix catcode regime in luatex sprint for `\pdf@shellescape` GH issue 45

[2018/09/10 v0.29]

- Actually do the fix described above in the code, not just document it.

[2019/07/25 v0.30]

- Remove uses of module function, see PR70

[2019/11/24 v0.31]

- Use iftex directly rather than ifluatex and ifpdf wrappers.
- detect `\filmoddate` and other XeTeX commands.
- Adjust `\pdf@escapestring` in LuaTeX to produce the same as in pdfTeX in the 8bit range and not drop all non ascii characters.

[2020-06-04 v0.32]

- Updated pdftexcmds.elapsedtime to lua 5.3 (issue #4).

[2020-06-24 v0.33]

- avoid that `\pdfelapsedtime` and `\pdfresettimer` are set to `\relax` when using xelatex (issue #5).
- load ltluatex when using plain so that the catcode tables are available.

[2026-10-28 v0.34]

- Add `\pdf@filedump` for XeTeX (issue #8).
- Update maintainer to the LaTeX Project.

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.

Symbols

`\&` 505, 507
`\@PackageError` 485
`\@PackageInfoNoLine` 294, 308, 313, 396, 400, 402
`\@PackageWarning` 445
`\@PackageWarningNoLine` 647
`\@ehc` 489
`\@nil` 508, 509, 511, 514, 518, 522, 530, 563, 572, 705
`\@undefined` 58, 293, 388
`\\` 836, 839

Numbers

`\0` 537, 690

A

`\aftergroup` 29

C

`\catcode` 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99, 505, 507, 537, 690

\catcodetable@string 673
\chardef 182, 184
\count 433, 435
\csname 14, 21, 50, 66, 76, 133, 136, 156, 157, 160, 162, 176, 194, 207, 209, 226, 227, 231, 232, 261, 266, 269, 270, 284, 286, 289, 310, 315, 321, 324, 330, 332, 341, 376, 493, 496, 623, 631, 635, 643

D

\delimiter 349, 355, 373
\directlua 240, 242, 540, 547, 552, 559, 568, 575, 580, 585, 590, 595, 600, 607, 612, 617, 672, 676, 681, 686
\draftmode 389

E

\empty 17, 18
\endcsname 14, 21, 50, 66, 76, 133, 136, 149, 157, 160, 162, 176, 194, 207, 209, 227, 231, 232, 261, 266, 269, 270, 284, 286, 289, 310, 315, 321, 324, 330, 332, 341, 376, 493, 496, 623, 631, 635, 643
\endinput 29, 129
\endlinechar 4, 35, 71, 77, 89, 246
\escapechar 128, 131, 265, 323
\everyjob 465, 466

F

\filedump 203

I

\ifcase 407, 440, 504
\ifcsname 149
\ifeof 180, 181
\ifluatex 148, 238, 282, 387, 451
\ifnum 180, 239, 288, 291, 349, 379, 414, 421, 458, 459, 499, 500, 535, 669, 691
\ifpdf 398
\ifx 15, 18, 21, 50, 58, 61, 133, 136, 156, 157, 162, 176, 194, 207, 209, 226, 227, 232, 261, 268, 284, 286, 289, 310, 315, 321, 329, 341, 356, 357, 360, 362, 388, 483, 493, 512, 523, 623, 631, 635, 643, 646
\ifxetex 201
\immediate 23, 52, 222
\input 137, 149

L

\ltx@backslashchar 396, 400, 403, 648, 650, 652, 653
\ltx@cclv 433, 435
\ltx@firstoftwo 350, 380, 415
\ltx@IfUndefined 178, 395
\ltx@one 399, 414, 421, 422
\ltx@onelevel@sanitize 475
\ltx@ReturnAfterElseFi 513, 524
\ltx@ReturnAfterFi 517, 529
\ltx@secondoftwo . . 352, 382, 409, 417
\ltx@zero 394, 408, 424, 525
\luaescapestring 541, 542, 548, 553, 561, 570, 576, 581, 586, 591, 596, 601, 602, 603, 608, 613, 618, 677, 703
\luatexversion 239, 458, 499, 500, 535, 669

M

\mdfivesum 212, 214
\meaning . . 264, 266, 324, 327, 343, 379
\MessageBreak 296, 446, 486, 487, 649, 651

N

\newlinechar 245, 246
\number 128, 601, 602, 673
\numexpr 634, 662, 672

P

\PackageInfo 26
\pdf@draftmode *5*, 408, 420
\pdf@elapsedtime *4*, 661
\pdf@escapehex *4*, 168, 546
\pdf@escapehexnative 168, 551
\pdf@escapename 579
\pdf@escapenamenative 584
\pdf@escapestring 574
\pdf@filedump *4*, 197, 202, 599
\pdf@filemdfivesum . . . *4*, 214, 219, 616
\pdf@filemoddate *4*, 594
\pdf@filesize *4*, 589
\pdf@ifdraftmode *5*, 409, 413
\pdf@ifprimitive *6*, 278, 314
\pdf@isprimitive *6*, 339, 342, 378, 392, 545
\pdf@lastsystemexit 685
\pdf@lastsystemstatus 680
\pdf@mdfivesum *4*, 212, 213, 217, 218, 606
\pdf@mdfivesumnative . . 213, 218, 611
\pdf@pipe *6*, 691
\pdf@primitive . . *6*, 274, 293, 295, 309
\pdf@resettimer *4*, 656
\pdf@setdraftmode *5*, 431, 446
\pdf@shellescape *5*, 182, 184, 189, 669
\pdf@strcmp *3*, 379, 539
\pdf@system *5*, 221, 675
\pdf@unescapehex *3*, 170, 556
\pdf@unescapehexnative . . *6*, 170, 565
\pdfdraftmode 388, 389, 412
\pdffiledump 198
\pdfmdfivesum 217, 219
\pdfprimitive 297
\pdfshellescape 190
\pdftexcmds@@isprimitive . . . 346, 348
\pdftexcmds@@setdraftmode . . 435, 439
\pdftexcmds@AtEnd 95, 96, 126, 127, 453, 708
\pdftexcmds@DecodeA 509, 511
\pdftexcmds@DecodeB 514, 522

\pdftexcmds@directlua 239, 247, 455, 460, 467, 477, 624, 636, 657, 663, 692, 701
\pdftexcmds@draftmode 412, 414, 421, 428
\pdftexcmds@equal 349, 355, 373
\pdftexcmds@equalcont 364, 370, 371, 376
\pdftexcmds@isprimitive . . . 343, 345
\pdftexcmds@nopdftex 151, 152, 158, 177, 195, 210, 228
\pdftexcmds@Patch 498, 504, 561, 570, 703
\pdftexcmds@PatchDecode 508, 557, 566, 699
\pdftexcmds@setdraftmode 410, 427, 441, 443
\pdftexcmds@strip@prefix . . . 258, 264
\pdftexcmds@temp 154, 166, 167, 169, 171, 172, 173, 174, 224, 235, 236, 259, 274, 275, 276, 277, 278, 279, 280, 281, 319, 337, 338, 394, 399, 407, 621, 632, 644, 646
\pdftexcmds@toks . . 492, 558, 567, 700
\pdftexrevision 291
\pdftexversion 180, 288
\protected 621, 656, 661, 671
\ProvidesPackage 19, 67

R

\RequirePackage 144, 145, 146
\romannumeral 557, 566, 699

S

\space 295, 297, 309, 314

T

\the 77, 78, 79, 80, 81, 82, 83, 84, 97, 435, 466, 558, 567, 700
\TMP@EnsureCode 94, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125
\TMP@RequirePackage 134, 140, 141, 142
\toksdef 494

W

\write 23, 52, 222

X

\x . . . 14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 263, 264, 268, 324, 329, 434, 437, 474, 475, 483, 487

Y

\y 266, 268, 325, 327, 329, 476, 483, 488

Z

\z 326, 327