The pdftexcmds package

Heiko Oberdiek*

2020-06-04 v0.32

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 primitive using Lua.

Contents

*Please report any issues at https://github.com/ho-tex/pdftexcmds/issues

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 easily simulated by macros. The primitives using key words such as `file` (`\pdfmdfivesum`) or `offset` and `length` (`\pdffiledump`) and uses ⟨*general text*⟩ for the other arguments. Using token registers assignments, ⟨*general text*⟩ could be catched. 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}
  ```

LuaTEX isn't stable yet and thus the status of this package is *experimental*. Feedback is welcome.

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. Exact 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.

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3   \catcode13=5 % ^^M
4   \endlinechar=13 %
5   \catcode35=6 % #
6   \catcode39=12 % '
7   \catcode44=12 % ,
8   \catcode45=12 % -
9   \catcode46=12 % .
10  \catcode58=12 % :
11  \catcode64=11 % @
12  \catcode123=1 % {
13  \catcode125=2 % }
14  \expandafter\let\expandafter\x\csname ver@pdftexcmds.sty\endcsname
15  \ifx\x\relax % plain-TeX, first loading
16  \else
17    \def\empty{}%
18    \ifx\x\empty % LaTeX, first loading,
19      % variable is initialized, but \ProvidesPackage not yet seen
20    \else
21      \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}%
  [2020-06-04 v0.32 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
\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

  \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
      \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{2020-06-04 v0.32}%
```

```
  \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("2020-06-04 v0.32")
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
```

```
735         elseif a == 194 then
736           table.insert(t, string.char(b))
737         elseif a == 195 then
738           table.insert(t, string.char(b + 64))
739         end
740       end
741     end
742   end
743   return table.concat(t)
744 end
745 function pdftexcmds.escapehex(str, mode)
746   if mode == "byte" then
747     str = utf8_to_byte(str)
748   end
749   tex.write((string.gsub(str, ".",
750     function (ch)
751       return string.format("%02X", string.byte(ch))
752     end
753   )))
754 end
```

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

```
755 function pdftexcmds.unescapehex(str, mode, patch)
756   local a = 0
757   local first = true
758   local result = {}
759   for i = 1, string.len(str), 1 do
760     local ch = string.byte(str, i)
761     if ch >= 48 and ch <= 57 then
762       ch = ch - 48
763     elseif ch >= 65 and ch <= 70 then
764       ch = ch - 55
765     elseif ch >= 97 and ch <= 102 then
766       ch = ch - 87
767     else
768       ch = nil
769     end
770     if ch then
771       if first then
772         a = ch * 16
773         first = false
774       else
775         table.insert(result, a + ch)
776         first = true
777       end
778     end
779   end
780   if not first then
781     table.insert(result, a)
782   end
783   if patch == 1 then
784     local temp = {}
785     for i, a in ipairs(result) do
786       if a == 0 then
787         table.insert(temp, 1)
788         table.insert(temp, 1)
789       else
790         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
```

```
894         timezone = "Z"
895       else
896         local hours = math.floor(off / 60)
897         local mins = math.abs(off - hours * 60)
898         timezone = string.format("%+03d'%02d'", hours, mins)
899       end
900       tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",
901           d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
902     end
903   end
904 end
905 function pdftexcmds.filedump(offset, length, filename)
906   length = tonumber(length)
907   if length and length > 0 then
908     local foundfile = kpse.find_file(filename, "tex", true)
909     if foundfile then
910       offset = tonumber(offset)
911       if not offset then
912         offset = 0
913       end
914       local filehandle = io.open(foundfile, "rb")
915       if filehandle then
916         if offset > 0 then
917           filehandle:seek("set", offset)
918         end
919         local dump = filehandle:read(length)
920         pdftexcmds.escapehex(dump)
921         filehandle:close()
922       end
923     end
924   end
925 end
926 function pdftexcmds.mdfivesum(str, mode)
927   if mode == "byte" then
928     str = utf8_to_byte(str)
929   end
930   pdftexcmds.escapehex(md5.sum(str))
931 end
932 function pdftexcmds.filemdfivesum(filename)
933   local foundfile = kpse.find_file(filename, "tex", true)
934   if foundfile then
935     local filehandle = io.open(foundfile, "rb")
936     if filehandle then
937       local contents = filehandle:read("*a")
938       pdftexcmds.escapehex(md5.sum(contents))
939       filehandle:close()
940     end
941   end
942 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)
    tex.write(result - 256 * math.floor(result / 256))
  end
end
function pdftexcmds.lastsystemexit()
  local result = tonumber(systemexitstatus)
  if result then
    tex.write(math.floor(result / 256))
  end
```

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

1036 ⟨/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.

Bundle. All the packages of the bundle 'pdftexcmds' 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/pdftexcmds.tds.zip

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

[1] CTAN:pkg/pdftexcmds

3.2 Bundle installation

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

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

Script installation. Check the directory `TDS:scripts/pdftexcmds/` for scripts that need further installation steps.

3.3 Package installation

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

```
tex 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 → scripts/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.

3.4 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`.

3.5 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).

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.