The pdftexcmds package

Heiko Oberdiek[*]
<heiko.oberdiek at googlemail.com>

2018/09/10 v0.29

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/oberdiek/issues

1 Documentation

Some primitives of pdfTEX [**pdftex-manual**] are not defined by LuaTEX [**luatex-manual**]. 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 [pdftex-manual]

`\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 [pdftex-manual]

`\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 [pdftex-manual]

The timekeeping macros are based on Andy Thomas' work [**AndyThomas:Analog**].

`\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`: $\geq 1\,\text{ms}$
- pdfTeX with `time`: $\geq 1\,\text{s}$
- LuaTeX: $\geq 10\,\text{ms}$
  (`os.clock()` returns a float number with two decimal digits in LuaTeX beta-0.70.1-2011061416 (rev 4277)).

1.2.4 Miscellaneous [pdftex-manual]

`\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 XƎTEX 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. XƎTEX 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

```
⟨*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}%
  [2018/09/10 v0.29 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{ifluatex}[2010/03/01]%
  \TMP@RequirePackage{ltxcmds}[2010/12/02]%
  \TMP@RequirePackage{ifpdf}[2010/09/13]%
\else
  \RequirePackage{infwarerr}[2007/09/09]%
  \RequirePackage{ifluatex}[2010/03/01]%
  \RequirePackage{ltxcmds}[2010/12/02]%
  \RequirePackage{ifpdf}[2010/09/13]%
\fi
```

2.4 Without LuaTEX

```
\ifluatex
\else
  \@PackageInfoNoLine{pdftexcmds}{LuaTeX not detected}%
  \def\pdftexcmds@nopdftex{%
    \@PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
    \let\pdftexcmds@nopdftex\relax
  }%
  \def\pdftexcmds@temp#1{%
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname pdf#1\endcsname\relax
      \pdftexcmds@nopdftex
    \else
      \expandafter\def\csname pdf@#1\expandafter\endcsname
      \expandafter##\expandafter{%
        \csname pdf#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 pdf#1\endcsname\relax
    \pdftexcmds@nopdftex
  \else
    \expandafter\let\csname pdf@#1\expandafter\endcsname
    \csname pdf#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%
\ifnum\luatexversion<80
  \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{luatex-loader}[2009/04/10]%
  \else
    \RequirePackage{luatex-loader}[2009/04/10]%
  \fi
\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{2018/09/10 v0.29}%
  \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 [pdftex-manual]

\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}", "byte")%
  }%
}
```

\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 [pdftex-manual]

\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 [pdftex-manual]

\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⟩
module("oberdiek.pdftexcmds", package.seeall)
local systemexitstatus
function getversion()
  tex.write("2018/09/10 v0.29")
end
```

2.11.1 Strings [pdftex-manual]

```
function 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 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 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,

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

See procedure escapestring in file utils.c of pdfTEX.

```
function 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 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 [pdftex-manual]

```
function 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 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 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)
        escapehex(dump)
        filehandle:close()
      end
    end
  end
end
function mdfivesum(str, mode)
  if mode == "byte" then
    str = utf8_to_byte(str)
  end
  escapehex(md5.sum(str))
end
function 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")
      escapehex(md5.sum(contents))
      filehandle:close()
    end
  end
end
```

2.11.3 Timekeeping [pdftex-manual]

The functions for timekeeping are based on Andy Thomas' work [**AndyThomas:Analog**]. 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 resettimer()
  basetime = os.clock()
end
function elapsedtime()
  local val = (os.clock() - basetime) * 65536 + .5
  if val > 2147483647 then
    val = 2147483647
  end
  tex.write(string.format("%d", val))
end
```

2.11.4 Miscellaneous [pdftex-manual]

```
function 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 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 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 lastsystemexit()
  local result = tonumber(systemexitstatus)
  if result then
    tex.write(math.floor(result / 256))
  end
end
function pipe(cmdline, patch)
  local result
  systemexitstatus = nil
  texio.write_nl("log", "pipe(" .. cmdline ..") ")
  if io.popen then
    texio.write("log", "executed.")
    local handle = io.popen(cmdline, "r")
    if handle then
      result = handle:read("*a")
      handle:close()
    end
  else
    texio.write("log", "disabled.")
  end
  if result then
    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
    tex.settoks(toks, result)
  else
    tex.settoks(toks, "")
  end
end
```

⟨/lua⟩

3 Test

3.1 Catcode checks for loading

```
⟨*test1⟩
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
  \countdef\count@=255 %
\fi
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
\expandafter\ifx\csname loop\endcsname\relax
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
{%
  \def\loop#1\repeat{%
    \def\body{#1}%
    \iterate
  }%
  \def\iterate{%
    \body
      \let\next\iterate
    \else
      \let\next\relax
    \fi
    \next
  }%
  \let\repeat=\fi
}%
\def\RestoreCatcodes{}
\count@=0 %
\loop
  \edef\RestoreCatcodes{%
    \RestoreCatcodes
    \catcode\the\count@=\the\catcode\count@\relax
  }%
\ifnum\count@<255 %
  \advance\count@ 1 %
\repeat

\def\RangeCatcodeInvalid#1#2{%
  \count@=#1\relax
  \loop
    \catcode\count@=15 %
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\def\RangeCatcodeCheck#1#2#3{%
  \count@=#1\relax
  \loop
    \ifnum#3=\catcode\count@
    \else
```

```
      \errmessage{%
        Character \the\count@\space
        with wrong catcode \the\catcode\count@\space
        instead of \number#3%
      }%
    \fi
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\def\space{ }
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input pdftexcmds.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\%=14 %
  \LoadCommand
  \RangeCatcodeCheck{0}{36}{15}%
  \RangeCatcodeCheck{37}{37}{14}%
  \RangeCatcodeCheck{38}{47}{15}%
  \RangeCatcodeCheck{48}{57}{12}%
  \RangeCatcodeCheck{58}{63}{15}%
  \RangeCatcodeCheck{64}{64}{12}%
  \RangeCatcodeCheck{65}{90}{11}%
  \RangeCatcodeCheck{91}{91}{15}%
  \RangeCatcodeCheck{92}{92}{0}%
  \RangeCatcodeCheck{93}{96}{15}%
  \RangeCatcodeCheck{97}{122}{11}%
  \RangeCatcodeCheck{123}{255}{15}%
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```

⟨/test1⟩

3.2 Test for \pdf@isprimitive

⟨*test2⟩

```
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\input pdftexcmds.sty\relax
\def\msg#1{%
  \begingroup
    \escapechar=92 %
    \immediate\write16{#1}%
  \endgroup
}
\long\def\test#1#2#3#4{%
  \begingroup
    #4%
    \def\str{%
      Test \string\pdf@isprimitive
      {\string #1}{\string #2}{...}: %
    }%
    \pdf@isprimitive{#1}{#2}{%
      \ifx#3Y%
        \msg{\str true ==> OK.}%
      \else
        \errmessage{\str false ==> FAILED}%
      \fi
    }{%
      \ifx#3Y%
        \errmessage{\str true ==> FAILED}%
      \else
        \msg{\str false ==> OK.}%
      \fi
    }%
  \endgroup
}
\test\relax\relax Y{}
\test\foobar\relax Y{\let\foobar\relax}
\test\foobar\relax N{}
\test\hbox\hbox Y{}
\test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
\test\if\if Y{}
\test\if\ifx N{}
\test\ifx\if N{}
\test\par\par Y{}
\test\hbox\par N{}
\test\par\hbox N{}
\csname @@end\endcsname\end
⟨/test2⟩
```

3.3 Test for \pdf@shellescape

```
⟨*test-shell⟩
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\input pdftexcmds.sty\relax
\def\msg#{\immediate\write16}
\def\MaybeEnd{}
\ifx\luatexversion\UnDeFiNeD
\else
  \ifnum\luatexversion<68 %
    \ifx\pdf@shellescape\@undefined
      \msg{SHELL=U}%
      \msg{OK (LuaTeX < 0.68)}%
    \else
      \msg{SHELL=defined}%
      \errmessage{Failed (LuaTeX < 0.68)}%
    \fi
    \def\MaybeEnd{\csname @@end\endcsname\end}%
  \fi
\fi
\MaybeEnd
\ifx\pdf@shellescape\@undefined
  \msg{SHELL=U}%
\else
  \msg{SHELL=\number\pdf@shellescape}%
\fi
\ifx\expected\@undefined
\else
  \ifx\expected\relax
    \msg{EXPECTED=U}%
    \ifx\pdf@shellescape\@undefined
```

```
      \msg{OK}%
    \else
      \errmessage{Failed}%
    \fi
  \else
    \msg{EXPECTED=\number\expected}%
    \ifnum\pdf@shellescape=\expected\relax
      \msg{OK}%
    \else
      \errmessage{Failed}%
    \fi
  \fi
\fi
\csname @@end\endcsname\end
⟨/test-shell⟩
```

3.4 Test for escape functions

```
⟨*test-escape⟩
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\^=7 %
\catcode`\@=11 %
\errorcontextlines=1000 %
\input pdftexcmds.sty\relax
\def\msg#1{%
  \begingroup
    \escapechar=92 %
    \immediate\write16{#1}%
  \endgroup
}

\begingroup
  \catcode`\@=11 %
  \countdef\count@=255 %
  \def\space{ }%
  \long\def\@whilenum#1\do #2{%
    \ifnum #1\relax
      #2\relax
      \@iwhilenum{#1\relax#2\relax}%
    \fi
  }%
  \long\def\@iwhilenum#1{%
    \ifnum #1%
      \expandafter\@iwhilenum
    \else
      \expandafter\ltx@gobble
    \fi
    {#1}%
  }%
  \gdef\AllBytes{}%
  \count@=0 %
  \catcode0=12 %
  \@whilenum\count@<256 \do{%
    \lccode0=\count@
    \ifnum\count@=32 %
      \xdef\AllBytes{\AllBytes\space}%
    \else
      \lowercase{%
        \xdef\AllBytes{\AllBytes^^@}%
      }%
    \fi
    \advance\count@ by 1 %
  }%
\endgroup
\def\AllBytesHex{%
  000102030405060708090A0B0C0D0E0F%
  101112131415161718191A1B1C1D1E1F%
  202122232425262728292A2B2C2D2E2F%
  303132333435363738393A3B3C3D3E3F%
  404142434445464748494A4B4C4D4E4F%
  505152535455565758595A5B5C5D5E5F%
  606162636465666768696A6B6C6D6E6F%
  707172737475767778797A7B7C7D7E7F%
  808182838485868788898A8B8C8D8E8F%
  909192939495969798999A9B9C9D9E9F%
  A0A1A2A3A4A5A6A7A8A9AAABACADAEAF%
  B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF%
  C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF%
  D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%
  E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
  F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
}
\ltx@onelevel@sanitize\AllBytesHex
\expandafter\lowercase\expandafter{%
  \expandafter\def\expandafter\AllBytesHexLC
      \expandafter{\AllBytesHex}%
}
\begingroup
  \catcode`\#=12 %
  \xdef\AllBytesName{%
    #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
    #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
    #20!"#23$#25&'#28#29*+,-.#2F%
    0123456789:;#3C=#3E?%
    @ABCDEFGHIJKLMNO%
    PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
    `abcdefghijklmno%
    pqrstuvwxyz#7B|#7D\string~#7F%
    #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
    #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
    #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
    #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
    #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
    #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
    #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
    #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
  }%
\endgroup
\ltx@onelevel@sanitize\AllBytesName
\edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
\begingroup
  \def\|{|}%
  \edef\%{\ltx@percentchar}%
  \catcode`\|=0 %
  \catcode`\#=12 %
  \catcode`\~=12 %
  \catcode`\\=12 %
  |xdef|AllBytesString{%
    \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
    \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
    \040!"#$|%&'\(\)*+,-./%
    0123456789:;<=>?%
    @ABCDEFGHIJKLMNO%
    PQRSTUVWXYZ[\\]^_%
    `abcdefghijklmno%
    pqrstuvwxyz{|}~\177%
    \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
    \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
    \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
    \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
    \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
    \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
    \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
    \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
  }%
|endgroup
\ltx@onelevel@sanitize\AllBytesString
\def\Test#1#2#3{%
  \begingroup
    \expandafter\expandafter\expandafter\def
    \expandafter\expandafter\expandafter\TestResult
    \expandafter\expandafter\expandafter{%
      #1{#2}%
    }%
    \ifx\TestResult#3%
    \else
      \newlinechar=10 %
      \msg{Expect:^^J#3}%
      \msg{Result:^^J\TestResult}%
      \errmessage{\string#2 -\string#1-> \string#3}%
    \fi
  \endgroup
}
\def\test#1#2#3{%
  \edef\TestFrom{#2}%
  \edef\TestExpect{#3}%
  \ltx@onelevel@sanitize\TestExpect
  \Test#1\TestFrom\TestExpect
}
\test\pdf@unescapehex{74657374}{test}
\begingroup
  \catcode0=12 %
  \catcode1=12 %
  \test\pdf@unescapehex{740074017400740174}{t^^@t^^At^^@t^^At}%
\endgroup
\Test\pdf@escapehex\AllBytes\AllBytesHex
\Test\pdf@unescapehex\AllBytesHex\AllBytes
\Test\pdf@escapename\AllBytes\AllBytesName
\Test\pdf@escapestring\AllBytes\AllBytesString
\csname @@end\endcsname\end
⟨/test-escape⟩
```

4 Installation

4.1 Download

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

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

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

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

[1] http://ctan.org/pkg/pdftexcmds

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

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

4.2 Bundle installation

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

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

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

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

4.3 Package installation

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

```
tex 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/oberdiek/pdftexcmds.sty
oberdiek.pdftexcmds.lua         → scripts/oberdiek/oberdiek.pdftexcmds.lua
pdftexcmds.lua                  → scripts/oberdiek/pdftexcmds.lua
pdftexcmds.pdf                  → doc/latex/oberdiek/pdftexcmds.pdf
test/pdftexcmds-test1.tex       → doc/latex/oberdiek/test/pdftexcmds-test1.tex
test/pdftexcmds-test2.tex       → doc/latex/oberdiek/test/pdftexcmds-test2.tex
test/pdftexcmds-test-shell.tex  → doc/latex/oberdiek/test/pdftexcmds-test-shell.tex
test/pdftexcmds-test-escape.tex → doc/latex/oberdiek/test/pdftexcmds-test-escape.tex
pdftexcmds.dtx                  → source/latex/oberdiek/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.

4.4 Refresh file name databases

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

4.5 Some details for the interested

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
```

5 Catalogue

The following XML file can be used as source for the TeX Catalogue. The elements `caption` and `description` are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is `pdftexcmds.xml`.

```
⟨*catalogue⟩
<?xml version='1.0' encoding='us-ascii'?>
<!DOCTYPE entry SYSTEM 'catalogue.dtd'>
<entry datestamp='$Date$' modifier='$Author$' id='pdftexcmds'>
  <name>pdftexcmds</name>
  <caption>LuaTeX support for pdfTeX utility functions.</caption>
  <authorref id='auth:oberdiek'/>
  <copyright owner='Heiko Oberdiek' year='2007,2009-2011'/>
  <license type='lppl1.3'/>
  <version number='0.20'/>
  <description>
    LuaTeX provides most of the commands of
    <xref refid='pdftex'>pdfTeX</xref> 1.40. However, a number of
    utility functions are not available.  This package tries to fill
    the gap and implements some of the missing primitives using Lua.
    <p/>
    The package is part of the <xref refid='oberdiek'>oberdiek</xref>
    bundle.
  </description>
  <documentation details='Package documentation'
      href='ctan:/macros/latex/contrib/oberdiek/pdftexcmds.pdf'/>
  <ctan file='true' path='/macros/latex/contrib/oberdiek/pdftexcmds.dtx'/>
  <miktex location='oberdiek'/>
  <texlive location='oberdiek'/>
  <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'/>
</entry>
⟨/catalogue⟩
```

6 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.

7 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.