The pdftexcmds package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/11/12 v0.2

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

1 Documentation

Some primitives of pdfTeX are not defined by LuaTeX. This package implements macro based solutions using Lua code for the following missing pdfTeX primitives;

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

- `\pdffilesize`
- `\pdffilemoddate`
- `\pdffiledump`
- `\pdfmdfivesum`
- `\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).

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

1.2 Macros

`\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.

`\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*⟩`}`.

`\pdf@shellescape`

Same as `\pdfshellescape`. It 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.

`\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?

1.2.1 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 documenta-

tion 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
3   \catcode44 12 % ,
4   \catcode45 12 % -
5   \catcode46 12 % .
6   \catcode58 12 % :
7   \catcode64 11 % @
8   \expandafter\let\expandafter\x\csname ver@pdftexcmds.sty\endcsname
9   \ifcase 0%
10    \ifx\x\relax % plain
11    \else
12      \ifx\x\empty % LaTeX
13      \else
14        1%
15      \fi
16    \fi
17  \else
18    \catcode35 6 % #
19    \catcode123 1 % {
20    \catcode125 2 % }
21    \expandafter\ifx\csname PackageInfo\endcsname\relax
22      \def\x#1#2{%
23        \immediate\write-1{Package #1 Info: #2.}%
24      }%
25    \else
26      \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27    \fi
28    \x{pdftexcmds}{The package is already loaded}%
29    \endgroup
30    \expandafter\endinput
31  \fi
32 \endgroup
```

Package identification:

```
33 \begingroup
34   \catcode35 6 % #
35   \catcode40 12 % (
36   \catcode41 12 % )
37   \catcode44 12 % ,
38   \catcode45 12 % -
39   \catcode46 12 % .
40   \catcode47 12 % /
41   \catcode58 12 % :
42   \catcode64 11 % @
43   \catcode123 1 % {
44   \catcode125 2 % }
45   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
46     \def\x#1#2#3[#4]{\endgroup
47       \immediate\write-1{Package: #3 #4}%
48       \xdef#1{#4}%
49     }%
50   \else
51     \def\x#1#2[#3]{\endgroup
```

```
      #2[{#3}]%
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@pdftexcmds.sty\endcsname
\ProvidesPackage{pdftexcmds}%
  [2007/11/12 v0.2 LuaTeX support for pdfTeX utility functions (HO)]
```

2.2 Catcodes

```
\begingroup
  \catcode123 1 % {
  \catcode125 2 % }
  \def\x{\endgroup
    \expandafter\edef\csname pdftexcmds@AtEnd\endcsname{%
      \catcode35 \the\catcode35\relax
      \catcode64 \the\catcode64\relax
      \catcode123 \the\catcode123\relax
      \catcode125 \the\catcode125\relax
    }%
  }%
\x
\catcode35 6 % #
\catcode64 11 % @
\catcode123 1 % {
\catcode125 2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\pdftexcmds@AtEnd{%
    \pdftexcmds@AtEnd
    \catcode#1 \the\catcode#1\relax
  }%
  \catcode#1 #2\relax
}
\TMP@EnsureCode{33}{12}% !
\TMP@EnsureCode{34}{12}% "
\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{61}{12}% =
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{95}{12}% _ (other)
\TMP@EnsureCode{126}{12}% ~ (other)
```

2.3 Load package infwarerr

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

2.4 Without LuaTeX

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
  \@PackageInfo{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
  \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
    \pdftexcmds@nopdftex
  \else
    \def\pdf@mdfivesum#{\pdfmdfivesum}%
    \let\pdf@mdfivesumnative\pdf@mdfivesum
    \def\pdf@filemdfivesum#{\pdfmdfivesum file}%
  \fi
  \def\pdf@system#{%
    \immediate\write18%
  }%
  \pdftexcmds@AtEnd
  \expandafter\endinput
\fi
```

2.5 Lua functions

\pdftexcmds@luastate

```
\chardef\pdftexcmds@luastate=0 %
```

\pdftexcmds@toks

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

\begingroup
  \escapechar=92 %
  \edef\\{\string\\}%
  \catcode35=12 % #
  \catcode37=12\relax
  \directlua\pdftexcmds@luastate{
    pdftexcmds = {}
    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
    function pdftexcmds.utf8_to_byte(str)
      local i = 0
      local n = string.len(str)
      local t = {}
      while i < n do
        i = i + 1
        local a = string.byte(str, i)
        if a < 128 then
          table.insert(t, string.char(a))
        else
          if a >= 192 and i < n then
            i = i + 1
            local b = string.byte(str, i)
            if b < 128 or b >= 192 then
              i = i - 1
            elseif a == 194 then
              table.insert(t, string.char(b))
            elseif a == 195 then
              table.insert(t, string.char(b + 64))
            end
          end
        end
      end
      return table.concat(t)
    end
    function pdftexcmds.escapehex(str, mode)
      if mode == "byte" then
        str = pdftexcmds.utf8_to_byte(str)
      end
      tex.write((string.gsub(str, ".",
        function (ch)
          return string.format("%02X", string.byte(ch))
        end
      )))
    end
```

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

```
    function pdftexcmds.unescapehex(str, mode)
```

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

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

```
  function pdftexcmds.filemoddate(filename)
    local foundfile = kpse.find_file(filename, "tex", true)
    if foundfile then
      local date = lfs.attributes(foundfile, "modification")
      if date then
        local d = os.date("*t", date)
        if d.sec >= 60 then
          d.sec = 59
        end
        local u = os.date("!*t", date)
        local off = 60 * (d.hour - u.hour) + d.min - u.min
        if d.year ~= u.year then
          if d.year > u.year then
            off = off + 1440
          else
            off = off - 1440
          end
        elseif d.yday ~= u.yday then
          if d.yday > u.yday then
            off = off + 1440
          else
```

```
            off = off - 1440
          end
        end
        local timezone
        if off == 0 then
          timezone = "Z"
        else
          local hours = math.floor(off / 60)
          local mins = math.abs(off - hours * 60)
          timezone = string.format("%+03d'%02d'", hours, mins)
        end
        tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",
            d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
      end
    end
  end
  function pdftexcmds.filedump(offset, length, filename)
    length = tonumber(length)
    if length and length > 0 then
      local foundfile = kpse.find_file(filename, "tex", true)
      if foundfile then
        offset = tonumber(offset)
        if not offset then
          offset = 0
        end
        local filehandle = io.open(foundfile, "r")
        if filehandle then
          if offset > 0 then
            filehandle:seek("set", offset)
          end
          local dump = filehandle:read(length)
          pdftexcmds.escapehex(dump)
        end
      end
    end
  end
  function pdftexcmds.mdfivesum(str, mode)
    if mode == "byte" then
      str = pdftexcmds.utf8_to_byte(str)
    end
    pdftexcmds.escapehex(md5.sum(str))
  end
  function pdftexcmds.filemdfivesum(filename)
    local foundfile = kpse.find_file(filename, "tex", true)
    if foundfile then
      local filehandle = io.open(foundfile, "r")
      if filehandle then
        local contents = filehandle:read("*a")
        pdftexcmds.escapehex(md5.sum(contents))
      end
    end
  end
  function pdftexcmds.shellescape()
    if os.execute then
      tex.write("1")
    else
      tex.write("0")
    end
  end
  function pdftexcmds.system(cmdline)
    pdftexcmds.systemexitstatus = nil
    texio.write_nl("log", "system(" .. cmdline .. ") ")
```

```
      if os.execute then
        texio.write("log", "executed.")
        pdftexcmds.systemexitstatus = os.execute(cmdline)
      else
        texio.write("log", "disabled.")
      end
    end
    function pdftexcmds.lastsystemstatus()
      local result = tonumber(pdftexcmds.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(pdftexcmds.systemexitstatus)
      if result then
        tex.write(math.floor(result / 256))
      end
    end
    function pdftexcmds.pipe(cmdline)
      local result
      pdftexcmds.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
        tex.settoks(pdftexcmds.toks, result)
      else
        tex.settoks(pdftexcmds.toks, "")
      end
    end
  }
\endgroup
```

\pdf@strcmp

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

\pdf@escapehex

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

\pdf@escapehexnative

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

```
  }%
}%
```

`\pdf@unescapehex`

```
\def\pdf@unescapehex#1{%
  \the\expandafter\pdftexcmds@toks
  \directlua\pdftexcmds@luastate{%
    pdftexcmds.toks="pdftexcmds@toks"%
    pdftexcmds.unescapehex("\luaescapestring{#1}", "byte")%
  }%
}%
```

`\pdf@unescapehexnative`

```
\def\pdf@unescapehexnative#1{%
  \the\expandafter\pdftexcmds@toks
  \directlua\pdftexcmds@luastate{%
    pdftexcmds.toks="pdftexcmds@toks"%
    pdftexcmds.unescapehex("\luaescapestring{#1}")%
  }%
}%
```

`\pdf@escapestring`

```
\long\def\pdf@escapestring#1{%
  \directlua\pdftexcmds@luastate{%
    pdftexcmds.escapestring("\luaescapestring{#1}", "byte")%
  }%
}
```

`\pdf@escapename`

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

`\pdf@escapenamenative`

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

`\pdf@filesize`

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

`\pdf@filemoddate`

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

`\pdf@filedump`

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

```
        "\luaescapestring{#3}")%
  }%
}%
```

`\pdf@mdfivesum`

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

`\pdf@mdfivesumnative`

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

`\pdf@filemdfivesum`

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

`\pdf@shellescape`

```
\def\pdf@shellescape{%
  \directlua\pdftexcmds@luastate{%
    pdftexcmds.shellescape()%
  }%
}
```

`\pdf@system`

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

`\pdf@lastsystemstatus`

```
\def\pdf@lastsystemstatus{%
  \directlua\pdftexcmds@luastate{%
    pdftexcmds.lastsystemstatus()%
  }%
}
```

`\pdf@lastsystemexit`

```
\def\pdf@lastsystemexit{%
  \directlua\pdftexcmds@luastate{%
    pdftexcmds.lastsystemexit()%
  }%
}
```

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

```
\ifnum0%
    \directlua\pdftexcmds@luastate{%
      if io.popen then %
        tex.write("1")%
      end%
    }%
    =1 %
```

```
  \def\pdf@pipe#1{%
    \the\expandafter\pdftexcmds@toks
    \directlua\pdftexcmds@luastate{%
      pdftexcmds.toks="pdftexcmds@toks"%
      pdftexcmds.pipe("\luaescapestring{#1}")%
    }%
  }%
\fi

\pdftexcmds@AtEnd
```
⟨/package⟩

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\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\{=1 %
  \catcode`\}=2 %
  \catcode`\#=6 %
  \catcode`\[=12 %
  \catcode`\]=12 %
  \catcode`\%=14 %
  \catcode`\ =10 %
  \catcode13=5 %
  \input pdftexcmds.sty\relax
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```

⟨/test1⟩

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.

CTAN:macros/latex/contrib/oberdiek/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):

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

```
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
pdftexcmds.pdf → doc/latex/oberdiek/pdftexcmds.pdf
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

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

```
pdftk pdftexcmds.pdf unpack_files output .
```

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

plain-TEX: Run docstrip and extract the files.

LATEX: Generate the documentation.

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

```
latex \let\install=y\input{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
makeindex -s gind.ist pdftexcmds.idx
pdflatex pdftexcmds.dtx
makeindex -s gind.ist pdftexcmds.idx
pdflatex pdftexcmds.dtx
```

5 History

[2007/11/11 v0.1]

- First version.

[2007/11/12 v0.2]

- Short description fixed.

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; numbers in roman refer to the code lines where the entry is used.

R

T

W

X