The ifpdf package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2009/04/10 v2.0

Abstract

This package looks for pdfTeX in pdf mode and implements and sets the switch `\ifpdf`. The detection is based on `\pdfoutput` and the package will not change this value. It works with plain or LaTeX formats.

Contents

1 Documentation

1.1 Introduction

It is commonly known that Hàn Thê Thành's pdfTEX generates PDF output directly and many people uses pdfTEX for this purpose. However the DVI output was never thrown away. In contrary, he new features for typesetting that works in both PDF and DVI mode.

In the meantime many TEX distributions replace the traditional TEX binary with pdfTEX. Then, for example, called as `latex` pdfTEX works in DVI mode with the LATEX format preloaded, called as `pdflatex` pdfTEX starts in PDF mode.

Often packages or users want to know, whether the current document is typset by pdfTEX in PDF mode, because the different modes have different capabilities (color setting, graphics inclusion, ...). For this purpose pdfTEX's `\pdfoutput` can be asked.

As regulary reader of TEX newsgroups and mailing lists I could observe many problems with this task. Common errors are:

- pdfTEX has *two* modes. Using pdfTEX does not mean that the user always want to have PDF mode. For example, the PostScript support is better in DVI mode in conjunction with a PostScript aware DVI driver (e.g. dvips). Also the additional typesetting features are mode independent and also available in DVI mode.
- LATEX's `\@ifundefined` inherited the side effect from `\csname`. Unknown commands are defined with the meaning of `\relax`. If it is checked, whether `\pdfoutput` is defined, then this should not be forgotten.
- Having `\pdfoutput` does not automatically mean PDF mode. Also the value of `\pdfoutput` must be asked.
- `\pdfoutput` must not be destroyed in some way. Later code and packages are fooled then and will perhaps make wrong decisions. For example they may drop support for PDF mode, because they do not know that pdfTEX is running at all.

Robin Fairbairns provides an entry for this topic in his excellent FAQ (`http://www.tex.ac.uk/faq`): Am I using PDFTeX?

1.2 Usage

The package ifpdf can be used with both plain-TEX and LATEX:

plain-TEX: `\input ifpdf.sty`

LATEX 2ε: `\usepackage{ifpdf}`

`\ifpdf` The package provides the switch `\ifpdf`:

```
\ifpdf
  ... do things, if pdfTeX is running in pdf mode ...
\else
  ... other TeX or pdfTeX in dvi mode ...
\fi
```

Users of the package ifthen can use the switch as boolean:

```
\boolean{pdf}
```

The package can also be used to set global documentclass options:

```
\RequirePackage{ifpdf}
\ifpdf
   \documentclass[pdftex,...]{...}
\else
   \documentclass[...]{...}
\fi
```

1.3 Specification

The package have the following properties:

- It asks the setting of `\pdfoutput` for detecting pdfTeX in PDF mode.
- It never changes `\pdfoutput`.
- If `\pdfoutput` is undefined or has the meaning `\relax`, but the engine provides the primitive `\pdfoutput`, then `\pdfoutput` is enabled or restored if possible (only LuaTeX, version 0.36.0 or higher).
- It can be used with many formats including plain-TeX and LaTeX.

The mode detection implements the following algorithm:

```
if undefined(\pdfoutput)
    \ifpdf := \iffalse % pdfTeXis not running
else
    if \pdfoutput ≤ 0
        \ifpdf := \iffalse % pdfTeX in DVI mode
    else
        \ifpdf := \iftrue % pdfTeX in PDF mode
    fi
fi
```

The function undefined checks both cases, undefined command and `\relax`.

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@ifpdf.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
```

```
      \def\x#1#2{%
        \immediate\write-1{Package #1 Info: #2.}%
      }%
    \else
      \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
    \fi
    \x{ifpdf}{The package is already loaded}%
    \endgroup
    \expandafter\endinput
  \fi
\endgroup
```

Package identification:

```
\begingroup
  \catcode35 6 % #
  \catcode40 12 % (
  \catcode41 12 % )
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode47 12 % /
  \catcode58 12 % :
  \catcode64 11 % @
  \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@ifpdf.sty\endcsname
\ProvidesPackage{ifpdf}%
  [2009/04/10 v2.0 Provides the ifpdf switch (HO)]
```

2.2 Catcodes

```
\begingroup
  \catcode123 1 % {
  \catcode125 2 % }
  \def\x{\endgroup
    \expandafter\edef\csname ifpdf@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\ifpdf@AtEnd{%
    \ifpdf@AtEnd
    \catcode#1 \the\catcode#1\relax
  }%
  \catcode#1 #2\relax
}
\TMP@EnsureCode{10}{12}% ^^J
\TMP@EnsureCode{39}{12}% '
\TMP@EnsureCode{40}{12}% (
\TMP@EnsureCode{41}{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{94}{7}% ^
\TMP@EnsureCode{96}{12}% `
```

2.3 Check for previously defined \ifpdf

```
\begingroup
  \expandafter\ifx\csname ifpdf\endcsname\relax
  \else
    \edef\i/{\expandafter\string\csname ifpdf\endcsname}%
    \expandafter\ifx\csname PackageError\endcsname\relax
      \def\x#1#2{%
        \edef\z{#2}%
        \expandafter\errhelp\expandafter{\z}%
        \errmessage{Package ifpdf Error: #1}%
      }%
      \def\y{^^J}%
      \newlinechar=10 %
    \else
      \def\x#1#2{%
        \PackageError{ifpdf}{#1}{#2}%
      }%
      \def\y{\MessageBreak}%
    \fi
    \x{Name clash, \i/ is already defined}{%
      Incompatible versions of \i/ can cause problems,\y
      therefore package loading is aborted.%
    }%
    \endgroup
    \ifpdf@AtEnd
    \expandafter\endinput
  \fi
\endgroup
```

2.4 \pdfoutput and LuaTeX

It might happen, that LuaTeX is running, but `\pdfoutput` does not exist. In version 0.40 only `\directlua` is available at startup time. The enabling Lua function was already added in version 0.36. Thus we can ignore older versions, here `\pdfoutput` is available at startup time.

```
\begingroup
  \expandafter\ifx\csname pdfoutput\endcsname\relax
  \else
    \def\skip#1\relax\endgroup{\csname fi\endcsname\endgroup}%
    \skip
  \fi
```

```
  \expandafter\ifx\csname directlua\endcsname\relax
    \def\skip#1\endgroup{\csname fi\endcsname\endgroup}%
    \skip
  \fi
  \expandafter\ifx\csname RequirePackage\endcsname\relax
    \input ifluatex.sty\relax
  \else
    \RequirePackage{ifluatex}[2009/04/10]%
  \fi
  \ifluatex
    \ifnum\luatexversion<36 %
```

Unhappily LuaTeX's `\primitive` (derived from pdfTeX's `\pdfprimitive`) cannot be used:

```
\protected\gdef\pdfoutput{\primitive\pdfoutput}
```

Setting a value works, but getting fails, because TeX does no longer see it as number. It is unexpandable and breaks numerical contexts.

```
    \else
      \directlua{tex.enableprimitives('ifpdf', {'pdfoutput'})}%
      \global\let\pdfoutput\ifpdfpdfoutput
    \fi
  \fi
  \relax
\endgroup
```

2.5 \ifpdf

`\ifpdf` Create and set the switch. `\newif` initializes the switch with `\iffalse`.

```
\newif\ifpdf
```

Test `\pdfoutput`. Is it defined and different from `\relax`? Someone could have used LaTeX internal `\@ifundefined`, or something else involving. Notice, `\csname` is executed inside a group for the test to cancel the side effect of `\csname`.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname pdfoutput\endcsname\relax
\else
  \ifnum\pdfoutput<1 %
```

`\pdfoutput=0` or negative, so not generating pdf.

```
  \else
    \pdftrue
  \fi
\fi
```

2.6 Test for fool attempts

```
\begingroup
  \expandafter\ifx\csname pdfoutput\endcsname\relax
  \else
    \escapechar=-1 %
    \edef\m{\meaning\pdfoutput}%
    \edef\p{%
      \string p\string d\string f%
      \string o\string u\string t\string p\string u\string t%
    }%
    \ifx\m\p
    \else
      \expandafter\ifx\csname PackageWarningNoLine\endcsname\relax
        \def\PackageWarningNoLine#1#2{%
          \immediate\write16{%
            Package `#1' Warning: #2.%
```

```
          }%
        }%
      \fi
      \PackageWarningNoLine{ifpdf}{%
        Someone has redefined \string\\pdfoutput%
      }%
    \fi
  \fi
\endgroup
```

2.7 Protocol entry

Log comment:

```
\begingroup
  \expandafter\ifx\csname PackageInfo\endcsname\relax
    \def\x#1#2{%
      \immediate\write-1{Package #1 Info: #2.}%
    }%
  \else
    \let\x\PackageInfo
    \expandafter\let\csname on@line\endcsname\empty
  \fi
  \x{ifpdf}{pdfTeX in pdf mode \ifpdf\else not \fi detected}%
\endgroup
\ifpdf@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
}
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input ifpdf.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\{=1 %
  \catcode`\}=2 %
  \catcode`\#=6 %
  \catcode`\[=12 %
  \catcode`\]=12 %
  \catcode`\%=14 %
  \catcode`\ =10 %
  \catcode13=5 %
  \LoadCommand
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```

⟨/test1⟩

4 Installation

4.1 Download

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

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

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

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

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

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

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

4.2 Bundle installation

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

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

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

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

4.3 Package installation

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

```
tex ifpdf.dtx
```

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

```
ifpdf.sty             → tex/generic/oberdiek/ifpdf.sty
ifpdf.pdf             → doc/latex/oberdiek/ifpdf.pdf
test/ifpdf-test1.tex → doc/latex/oberdiek/test/ifpdf-test1.tex
ifpdf.dtx             → source/latex/oberdiek/ifpdf.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 ifpdf.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{ifpdf.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 ifpdf.dtx
makeindex -s gind.ist ifpdf.idx
pdflatex ifpdf.dtx
makeindex -s gind.ist ifpdf.idx
pdflatex ifpdf.dtx
```

5 History

[2001/06/14 v1.0]

- First public version.

[2001/07/14 v1.1]

- Documentation addition: global options

[2001/09/26 v1.2]

- Documentation typo corrected.
- Version number corrected.
- Line number in log entry removed.

[2005/07/22 v1.3]

- Some source code comments from Robin Fairbairns added.
- Bug fix for negative values of `\pdfoutput` (Oleg Katsitadze)
- LPPL 1.3
- Installation section with locations added.

[2006/02/20 v1.4]

- DTX framework.
- More robust check in case of undefined `\pdfoutput`.
- Extended documentation.

[2007/09/09 v1.5]

- Catcode settings added.

[2007/12/12 v1.6]

- Minor update.

[2008/12/12 v1.7]

- Fix in documentation for `\boolean` (found by S. Venkataraman).
- Code is not changed.

[2009/04/10 v2.0]

- Support for LuaTeX 0.40 added.
- Checks, whether `\pdfoutput` was changed.

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.