filedate.sty

—

Access and Compare Info and Modification Date*

Uwe Lück†

October 17, 2012

Abstract

filedate.sty provides basic access to the date of a LaTeX source file according to its `\ProvidesFile`, `\ProvidesPackage`, or `\ProvidesClass` entry—the "info date"—, as well as to its modification date according to `\pdffilemoddate` if the latter is available. Moreover commands are provided to compare the "info date" with the modification date, with "today"'s date, or with another date—that a script accessing modification dates such as adhocfilelist.sh may insert—, and to choose the effect of comparisons (error vs. "notice," reference date characterization). Thus updating the "info date" ("**date consistency**") of a source file may be ensured by a test during typesetting from it or by some (shell/TeX) script.

Related packages:: filemod, getfiledate, zwgetfdate, fileinfo

Keywords: package management, document versions

Contents

*This document describes version v0.2 of filedate.sty as of 2012/10/17.

†http://contact-ednotes.sty.de.vu

1 Features and Usage

1.1 Installing and Calling

The file filedate.sty is provided ready, installation only requires putting it somewhere where TEX finds it (which may need updating the filename data base).[1]

Below the `\documentclass` line(s) and above `\begin{document}`, you load filedate.sty (as usually) by

`\usepackage{filedate}`

but in "TEX scripts" such as below,

`\RequirePackage{filedate}`

is better.

1.2 Demonstration with a "TEX script" Example

The accompanying `wrong.tex` is an example of a "filedate TEX script" demonstrating what may go wrong.

```
\ProvidesFile{wrong.tex}[2012/10/15 filedate.sty demo]
\RequirePackage{filedate}
\CheckDateOfPDFmod{wrong}
\CheckDateOfPDFmod{wrong.tex}
\CheckDateOfToday{wrong.tex}
\stop
```

You may run it (by the command line '`latex wrong`') and experience:

1. `wrong.tex`'s "info date" is '`2012/10/15`', but its modification date is at least one day later.

2. `\CheckDateOfPDFmod{wrong}` demonstrates that in

   `\CheckDateOfPDFmod{`⟨*file*⟩`}`

   ⟨*file*⟩ must be the filename *including extension.* Otherwise the "info date" may be (displayed as) "unknown."

[1] `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf`

3. `\CheckDateOfPDFmod{wrong.tex}` tests against `wrong.tex`'s modification date according to `\pdffilemoddate`—the present package documentation uses pdftex indeed.

4. `\CheckDateOfToday{wrong.tex}` tests against "today"'s date, which should be different from `2012/10/15`.

5. The "script" terminates on LaTeX's `\stop` command, without typesetting anything. TeX is just used as a program, a command interpreter (as with docstrip).

2 Implementation and Single Commands

2.1 Package File Header (Legalese)

```
1  \NeedsTeXFormat{LaTeX2e}[1994/12/01]
2  \ProvidesPackage{filedate}[2012/10/17 v0.2 check file dates (UL)]
3
4  %% Copyright (C) 2012 Uwe Lueck,
5  %% http://www.contact-ednotes.sty.de.vu
6  %% -- author-maintained in the sense of LPPL below --
7  %%
8  %% This file can be redistributed and/or modified under
9  %% the terms of the LaTeX Project Public License; either
10 %% version 1.3c of the License, or any later version.
11 %% The latest version of this license is in
12 %%     http://www.latex-project.org/lppl.txt
13 %% We did our best to help you, but there is NO WARRANTY.
14 %%
15 %% Please report bugs, problems, and suggestions via
16 %%
17 %%   http://www.contact-ednotes.sty.de.vu
18 %%
```

2.2 The readprov Package

```
19 % \RequirePackage{readprov}
```

—is required for `\ReadInfoDate{`⟨*file*⟩`}` and `\ReadCheckDateOf{`⟨*file*⟩`}` (sections 2.3 and 2.6) only. Please care for providing it on your own if you need that.

2.3 Accessing "Info Date"

`\theinfodateof{`⟨*file*⟩`}` will expand to the first "word" of the `\Provides...` entry, provided that has been read before:

```
\newcommand*{\theinfodateof}[1]{%
    \@ifundefined{ver@#1}{unknown}{%
        \expandafter\expandafter\expandafter
            \fd@firstword\csname ver@#1\endcsname\@gobble{} \@nil}}
\def\fd@firstword#1 #2\@nil{#1}
```

This avoids the `\relax` that `\UseDateOf` from readprov currently adds (which doesn't harm in printing but is bad for comparing).

`\LoadInfoDateOf{`⟨*file*⟩`}` sets `\theinfodate` to the first word of what is in the `\Provides` instruction of ⟨*file*⟩, provided that info has been input. So far, you must care for yourself that this works.

```
\newcommand*{\LoadInfoDateOf}[1]{%
    \edef\theinfodate{\theinfodateof{#1}}}
```

`\ReadInfoDateOf{`⟨*file*⟩`}` additionally inputs the info before:

```
\newcommand*{\ReadInfoDateOf}[1]{%
    \ReadFileInfos{#1}\LoadInfoDateOf{#1}}
```

TODO provide automatically.

2.4 Accessing `\pdffilemoddate`

`\pdffilemoddate{`⟨*file*⟩`}` in the first instance is a pdftex primitive. With LuaTEX, pdftexcmds provides it. Currently, you must care for this yourself before loading the present package. I recommend the filemod documentation for details about `\pdffilemoddate`.

Otherwise, (with XELATEX) the modification date may be obtained by a (shell) script—the next definitions do *not* deal with the latter situation. Testing against "today" (`\rawtoday` in Section 2.5) may be another alternative.

`\thepdfmoddateof{`⟨*file*⟩`}` expands to the modification date (eight digits separated by two slashes) if `\pdffilemoddate` is available. Otherwise, we are trying to inform about unavailability:

```
\ifx\pdffilemoddate\@undefined
    \newcommand*{\thepdfmoddateof}{%
        \string\pdffilemoddate\space unavailable.}
\else
    \newcommand*{\thepdfmoddateof}[1]{%
        \expandafter \fd@pdftexdate \pdffilemoddate{#1}\@nil}
    \expandafter \def \expandafter
        \fd@pdftexdate\string D:#1#2#3#4#5#6#7#8#9\@nil{%
            #1#2#3#4/#5#6/#7#8}
```

—cf. Will Robertson's suggestion dating from 2010 on `stackoverflow.com` in another discussion of accessing modification dates, including use of scripts. `\string␣D` deals with the fact that `\pdffilemoddate` returns in "other" character tokens.

```
\fi
```

2.5 \rawtoday

`\rawtoday` accesses "today"'s date as eight digits separated by two slashes (`yyyy/mm/dd`):

```
\newcommand*{\rawtoday}{%
    \the\year/\two@digits{\the\month}/\two@digits{\the\day}}
```

2.6 Comparing

`\CheckDateOf{`⟨*file*⟩`}{`⟨*date*⟩`}` compares ⟨*file*⟩'s info date with ⟨*date*⟩:

```
\newcommand*{\CheckDateOf}[2]{%
```

We provide a check that does not affect the order with myfilist.

```
%     \ReadFileInfos{#1}%
```

The date according to `\Provides` will be accessible as `\theinfodate`:

```
    \LoadInfoDateOf{#1}%
%     \show\theinfodate
    \ReadPDFmodDateOf{#1}%
    \edef\@tempb{#2}%
%     \show\@tempb
    \ifx\theinfodate\@tempb
        \fd@datesequal{#1}%
    \else
        \fd@datesdiff{#1}%
    \fi}
```

`\ReadCheckDateOf{`⟨*file*⟩`}{`⟨*date*⟩`}` prepends `\ReadFileInfos{`⟨*file*⟩`}` from the readprov package (cf. Section 2.2):

```
\newcommand*{\ReadCheckDateOf}[1]{%
    \ReadFileInfos{#1}\CheckDateOf{#1}}
```

TODO provide automatically.

2.7 Reporting Styles

By default, there is no report about comparisons finding equality.

```
\let\fd@datesequal\@gobble
```

We do not want to disturb `\listfiles` with myfilist. `\EqualityMessages` changes this to screen and log messages:

```
\newcommand*{\EqualityMessages}{\let\fd@datesequal\fd@equalmess}
\def\fd@equalmess#1{\message{ + #1 passed date check + }}
\def\fd@errdatesdiff#1{%
    \PackageError{filedate}{%
        \fd@infodate{#1}\fd@datekind\@tempb}{%
        Fix that!}}
```

`\fd@infodate{`⟨*file*⟩`}` might be used to change the current presentation of the "info date:"

```
\def\fd@infodate#1{%
    #1 has \string\Provides... date \theinfodate\space}
```

TODO here `\theinfodate` could be replaced by `\theinfodateof{#1}`, there is no essential application of `\theinfodate` currently.

After `\DatesDiffErrors`, date differences are reportet "drastically" by `\PackageError`:

```
\newcommand*{\DatesDiffErrors}{\let\fd@datesdiff\fd@errdatesdiff}
```

This is the default:

```
\DatesDiffErrors
```

After `\DatesDiffNotices`, date differences are told by `\typeout`:

```
\newcommand*{\DatesDiffNotices}{\let\fd@datesdiff\fd@notedatesdiff}
\def\fd@notedatesdiff#1{\typeout{\fd@infodate{#1}}}
```

After `\ModDates`, reference dates are called "modification" dates:

```
\newcommand*{\ModDates}{\let\fd@datekind\fd@moddate}
\def\fd@moddate#1{\MessageBreak vs. modification date #1}
```

After `\SomeDates`, the type of reference dates is not specified. This is more accurate when the info date is compared with `\rawtoday`.

```
\newcommand*{\SomeDates}{\let\fd@datekind\fd@somedate}
\def\fd@somedate#1{\MessageBreak vs. #1}
```

That's the default:

```
\SomeDates
```

2.8 Reference Date "Types"

`\ReadPDFmodDateOf{`⟨*file*⟩`}` enables

`\CheckDateOf{`⟨*file*⟩`}{\thepdfmoddate}`

by setting `\thepdfmoddate`:

```
\newcommand*{\ReadPDFmodDateOf}[1]{%
    \edef\thepdfmoddate{\thepdfmoddateof{#1}}}
```

After a single `\UseReferenceDate{`⟨*date*⟩`}` all ensuing

`\CheckDateOfGiven{`⟨*file*⟩`}`

compare ⟨*file*⟩'s "info date" with ⟨*date*⟩. The latter may be an explicit

⟨*4-digits*⟩`/`⟨*2-digits*⟩`/`⟨*2-digits*⟩ (`yyyy/mm/dd`)

–a script might insert it—, `\rawtoday`, or `\thepdfmoddate`. adhocfilelist v0.7 (with option `-c`) is such a script, a shell script generating a "TEX script", providing the file modification date according to Unix/Linux.

```
\newcommand*{\UseReferenceDate}{\def\thedategiven}
\newcommand*{\CheckDateOfGiven}[1]{\CheckDateOf{#1}{\thedategiven}}
```

`\CheckDateOfPDFmod{`⟨*file*⟩`}` compares the "info date" with the modification date according to `\pdffilemoddate`, and in reporting a difference the modification date it is called a "modification date" indeed:

```
\newcommand*{\CheckDateOfPDFmod}[1]{%
    \begingroup
        \ModDates
        \CheckDateOf{#1}{\thepdfmoddate}%
    \endgroup}
```

`\CheckDateOfToday{`⟨*file*⟩`}` checks if the `\Provides` date is today's, and the report of a difference somewhat emphasizes that this may not be a *modification* date. (It may be a *substitute* for a modification date when you know that the file was modified "today".)

```
\newcommand*{\CheckDateOfToday}[1]{%
    \begingroup
        \def\fd@datekind ##1{%
            \MessageBreak which is not today}%
        \CheckDateOf{#1}{\rawtoday}%
    \endgroup}
```

TODO single check commands for comma-separated list of files, (dowith) automatic tests by `\ReadFileInfos` etc. (fileinfo).

2.9 Leaving the Package File

```
\endinput
```

2.10 VERSION HISTORY

```
v0.1   2012/10/15   core try, bad
v0.2   2012/10/16   code for first release
       2012/10/17   reordering, correcting documentation
```

3 Use with Present Package Documentation

At this place, the documentation source '`filedate.tex`' issues

```
\EqualityMessages
\CheckDateOfPDFmod{filedate.sty}
\CheckDateOfPDFmod{filedate.tex}
```

in order to check whether the "info dates" of the package file '`filedate.sty`' and of the documentation source and driver '`filedate.tex`' are the same as their modification dates according to `\pdffilemoddate` (using `pdflatex`). When I added this on 2012-10-17, it indeed informed me that I had not updated `filedate.tex`'s info date (`2012/10/16`, generation of first version of the file from a template, draft).

`\EqualityMessages` confirms that the tests were run indeed.