filedate.sty

—

Access and Compare LaTeX File Info and Modification Date*

Uwe Lück†

March 26, 2013

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. v0.4 enables checking info dates automatically as soon as a LaTeX file is loaded while typesetting or in a myfilist script.

Related packages:: filemod, getfiledate, zwgetfdate, fileinfo

Keywords: modification date, metadata, package documentation, document versions, macro programming

Contents

1 Features and Usage **2**
1.1 Basics of Usage 2
1.1.1 The Most Interesting Command 2
1.1.2 Installing and Calling 2
1.1.3 Demonstration with a "TeX script" Example 3

*This document describes version v0.41 of filedate.sty as of 2013/03/26.

†`http://contact-ednotes.sty.de.vu`

2 Implementation and Single Commands 4
2.1 Package File Header (Legalese) 4
2.2 The readprov Package 4
2.3 Providing Dates for Comparisons 4
2.3.1 Accessing "Info Date" 4
2.3.2 Accessing \pdffilemoddate 5
2.3.3 \rawtoday 5
2.4 Comparing Basically 6
2.5 Reporting Styles 6
2.5.1 Same Dates 6
2.5.2 Differing Dates 7
2.5.3 Kinds of Reference Dates 7
2.6 Shorthands 8
2.6.1 Kinds of Reference Dates 8
2.6.2 Automatic Immediate Checking 9
2.6.3 Level-Restricted Automatic Checking 10
2.6.4 User-Restricted Automatic Checking 10
2.7 Leaving the Package File 11
2.8 VERSION HISTORY 12

3 Use with Present Package Documentation 13

1 Features and Usage

1.1 Basics of Usage

1.1.1 The Most Interesting Command

The package allows to check whether the file **info** date ⟨*date*⟩ according to `\Provides` near the top of a LaTeX input file ⟨*file*⟩—i.e.,

`\Provides...{`⟨*file*⟩`}[`⟨*date*⟩␣`...]`

has been updated the same day when ⟨*file*⟩ actually was **modified** most recently. With pdfTeX, this can be checked by

`\CheckDateOfPDFmod{`⟨*file*⟩`}`

1.1.2 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}`

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

but in "TEX scripts" such as below,

```
\RequirePackage{filedate}
```

is better.

1.1.3 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."
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)

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{filedate}[2013/03/26 v0.41 check file dates (UL)]

%% Copyright (C) 2012 2013 Uwe Lueck,
%% http://www.contact-ednotes.sty.de.vu
%% -- author-maintained in the sense of LPPL below --
%%
%% This file can be redistributed and/or modified under
%% the terms of the LaTeX Project Public License; either
%% version 1.3c of the License, or any later version.
%% The latest version of this license is in
%%     http://www.latex-project.org/lppl.txt
%% We did our best to help you, but there is NO WARRANTY.
%%
%% Please report bugs, problems, and suggestions via
%%
%%   http://www.contact-ednotes.sty.de.vu
%%
```

2.2 The readprov Package

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

—is required for

\ReadInfoDate{⟨*file*⟩} and \ReadCheckDateOf{⟨*file*⟩}{⟨*date*⟩}

(sections 2.3.1 and 2.4) only. Please care for providing readprov [2] on your own if you need that.

2.3 Providing Dates for Comparisons

2.3.1 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).

[2] http://www.ctan.org/pkg/readprov

`\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.

2.3.2 Accessing `\pdffilemoddate`

`\pdffilemoddate{`⟨*file*⟩`}` in the first instance is a pdftex primitive. With LuaTEX, pdftexcmds[3] provides it. Currently, you must care for this yourself before loading the present package. I recommend the filemod[4] documentation for details about `\pdffilemoddate`.

`\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 "other" character tokens.

```
\fi
```

The modification date of ⟨*file*⟩ according to `\pdffilemoddate` will be available as `\thepdffilemoddate` after `ReadPDFfileModDateOf{`⟨*file*⟩`}`, see Section 2.6.1.

2.3.3 `\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}}
```

[3] `http://www.ctan.org/pkg/pdftexcmds`
[4] `http://www.ctan.org/pkg/filemod`

2.4 Comparing Basically

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

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

We provide a check that does not affect the order with myfilist [5].

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

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

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

The **reference** date may be either (i) *today* as accessed by `\rawtoday` (Section 2.3.3), (ii) the *modification* date as accessed by `\pdffilemoddate` (Section 2.3.2), (iii) something *else* relevant, e.g., a modification date determined and inserted by a shell script.

`\ReadCheckDateOf{⟨file⟩}{⟨date⟩}` prepends `\ReadFileInfos{⟨file⟩}` from the readprov package (cf. Section 2.2), in order to ensure that the **info** date is known:

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

TODO provide automatically.

2.5 Reporting Styles

2.5.1 Same Dates

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

```
\let\FD@datesequal\@gobble
```

`\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@refdate}{%      %% \fd@refdate 2012/10/19
        Fix that!}}
```

[5] http://www.ctan.org/pkg/myfilist

`\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.

2.5.2 Differing Dates

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 reported by `\typeout`:

```
\newcommand*{\DatesDiffNotices}{\let\FD@datesdiff\FD@notedatesdiff}
\def\FD@notedatesdiff#1{\def\MessageBreak{^^J}%   %% added 2012/10/24
                        {\typeout{\FD@infodate{#1}%
                                  \FD@refdate}}}  %% added 2012/10/19
```

v0.3 adds `\DatesDiffWarnings` to get more salient reports of date differences by `\PackageWarningNoLine`:

```
\newcommand*{\DatesDiffWarnings}{\let\FD@datesdiff\FD@warndatesdiff}
\def\FD@warndatesdiff#1{%
    \PackageWarningNoLine{filedate}%
                         {\FD@infodate{#1}\FD@refdate}}
```

2.5.3 Kinds of Reference Dates

When the reference date is `\pdffilemoddate`, the report about a comparison may call it a "modification date". But when the reference date is "today", it may not be a "modification date". Otherwise, it depends ... (See Section 2.4 for kinds of reference dates.)

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

```
\newcommand*{\ModDates}{\let\FD@refdate\FD@moddate}
\def\FD@moddate{\MessageBreak vs. modification date \FD@therefdate}
```

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@refdate\FD@somedate}
\def\FD@somedate{\MessageBreak vs. \FD@therefdate}
```

That's the default:

```
\SomeDates
```

2.6 Shorthands

We may want to compare *info* with *reference* dates for many files. But when the *reference* date is `\pdffilemoddate` for many files (probably the most frequent application, see Section 2.3.2), we don't want to state this explicitly for each file—Section 2.6.1—neither for other kinds of reference dates.

As to enumerating *file names*, the filesdo package from the commado [6] bundle should help (combinations of base names and extensions, see Section 3; TODO: shorthands using filesdo could be defined in the present package).

The files ⟨*in-file*⟩ whose dates we want to check may be just the same that some ⟨*reading-file*⟩ tries to `\input`. It may me helpful if those input commands trigger the file consistency check without a need to demand this explicitly for each file—the section on "automatic immediate checking" (Section 2.6.2) aims at this.

2.6.1 Kinds of Reference Dates

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

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

by setting `\thepdfmoddate`:

```
79  \newcommand*{\ReadPDFmodDateOf}[1]{%
80      \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 [7] 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.

```
81  \newcommand*{\UseReferenceDate}{\def\thedategiven}
82  \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:

```
83  \newcommand*{\CheckDateOfPDFmod}[1]{%
84      \begingroup
85          \ModDates
86          \CheckDateOf{#1}{\thepdfmoddate}%
87      \endgroup}
```

[6] http://www.ctan.org/pkg/commado
[7] http://www.ctan.org/pkg/adhocfilelist

`\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@refdate{%
            \MessageBreak which is not today}%
        \CheckDateOf{#1}{\rawtoday}%
    \endgroup}
```

2.6.2 Automatic Immediate Checking

`\FileDateAutoChecks` modifies the '`\Provides...`' commands internally so that they check their date with with the file's modification date according to `\pdffilemoddate`—this choice might be queried (TODO). The modification may be undone by `\NoFileDateAutoChecks` so that the '`\Provides...`' commands get their original functionality (only) again. This will even work with readprov v0.4.

Actually, LaTeX's internal `\@pr@videpackage` and `\@profidesfile` are used as hooks. Their original meanings are stored so they can be regained by '`\NoFile...`':

```
\let\FD@@provpkg\@pr@videpackage
\def\FD@provpkg[#1]{\FD@@provpkg[#1]%
                    \CheckDateOfPDFmod{\@currname.\@currext}}
\let\FD@@provfile\@providesfile
\def\FD@provfile#1[#2]{\FD@@provfile{#1}[#2]%
                       \CheckDateOfPDFmod{#1}}
```

`\FileDateAutoChecks*` in addition to `\FileDateAutoChecks` checks the main file's info date, assuming the main file is `\jobname.tex` (TODO):

```
\newcommand*{\FileDateAutoChecks}{%
    \@ifstar{{\FD@check@pdf@job\FD@autochecks}\FD@autochecks}%
                                            \FD@autochecks}
```

`\FD@check@pdf@job` is introduced in v0.41 for the "restricted" automatic checks below.

```
\newcommand*{\FD@check@pdf@job}{\CheckDateOfPDFmod{\jobname.tex}}
\newcommand*{\FD@autochecks}{\let\@pr@videpackage\FD@provpkg
                             \let\@providesfile\FD@provfile}
\newcommand*{\NoFileDateAutoChecks}{\let\@pr@videpackage\FD@@provpkg
                                    \let\@providesfile  \FD@@provfile}
```

However, see Section 2.6.3 for **problems** of the previous idea (unless using fileinfo[8], i.e., readprov or even myfilist).

[8] http://www.ctan.org/pkg/fileinfo

2.6.3 Level-Restricted Automatic Checking

The present package is made to ensure "date consistency" of files that the user maintains/edits. By contrast, date consistency cannot be expected for LaTeX package files that were generated by docstrip [9] (many days after the `.dtx` source was released). No date consistency checks should be applied to such files. (The "criticism" of the present section does not apply to usage with fileinfo, and the commands presented here *disable* fileinfo, so don't use them with fileinfo.)

The problem with `\FileDateAutoChecks` (Section 2.6.2) is, e.g., that a chapter file for a book that is `\included` by the master file may trigger loading a font definition file (`.fd`)—that is not date-consistent, though the test is applied to it. We now aim at checking only those files that are input by the file that contains the call, not to files that are input indirectly. We use `\InputIfFileExists` as a hook for this. This should affect *LaTeX* user commands for file inclusion, even for (the user's) packages and class files. It does *not* affect the version of `\input` without braces. The commands for this purpose are `\FileDateLevelChecks`, `\FileDateLevelChecks*`, and `\NoFileDateLevelChecks` (replacing `Auto` by `Level` for analogy to the commands of Section 2.6.2).

The implementation is somewhat recursive, or "counter-recursive"?

```
\def\FD@level@provpkg   [#1]{\FD@provpkg[#1]%
                               \NoFileDateLevelChecks}
\def\FD@level@provfile#1[#2]{\FD@provfile{#1}[#2]%
                               \NoFileDateLevelChecks}
```

`\FD@userchecks@` is introduced for Section 2.6.4:

```
\newcommand*{\FD@userchecks@}{\let\@pr@videpackage\FD@level@provpkg
                              \let\@providesfile\FD@level@provfile}
\newcommand*{\FD@levelchecks}{\FD@userchecks@
                              \let\InputIfFileExists\FD@input@if@f@ex}
\let\FD@@input@if@f@ex\InputIfFileExists
\newcommand{\FD@input@if@f@ex}[3]{\FD@@input@if@f@ex{#1}{#2}{#3}%
                                  \FD@levelchecks}
\newcommand*{\FileDateLevelChecks}{%
    \@ifstar{\CheckDateOfPDFmod{\jobname.tex}\FD@levelchecks}%
                                            \FD@levelchecks}
\newcommand*{\NoFileDateLevelChecks}{%
    \NoFileDateAutoChecks \let\InputIfFileExists\FD@@input@if@f@ex}
```

Section 2.6.4 introduces another idea to avoid checking of LaTeX package files.

2.6.4 User-Restricted Automatic Checking

Another idea to avoid checking of LaTeX package files while checking `\input` files automatically are `\FileDateUserChecks` and `\FileDateUserChecks*` only affecting user transclusion commands `\include{`⟨*file*⟩`}` and `\input{`⟨*file*⟩`}` while allowing nesting, i.e., e.g., even an `\input{`⟨*indir*⟩`}` in an `\included` file is

[9] `http://www.ctan.org/pkg/docstrip`

checked automatically. (See Section 2.6.3 if you do not remember what is bad with Section 2.6.2.)

```
\let\FD@@iinput\@iinput \let\FD@@include\@include
```

Using definitions from above:

```
\newcommand*{\FD@userchecks}{%
```

`\@iinput` is LaTeX's internal behind `\input`:

```
    \def\@iinput##1{%
        \FD@userchecks@\FD@@iinput{##1}\NoFileDateAutoChecks}%
```

`\@include` is LaTeX's internal behind `\include`. Its argument is limited by a space:

```
    \def\@include##1 {%
        \FD@userchecks@\FD@@include##1 \NoFileDateAutoChecks}}
```

Temporary switching off `\FD@provpkg` and `\FD@provfile` is doubled to deal with files that don't have a `\Provides` entry.

```
\newcommand*{\FileDateUserChecks}{%
    \@ifstar{\FD@check@pdf@job\FD@userchecks}\FD@userchecks}
\newcommand*{\NoFileDateUserChecks}{%
    \let\@iinput\FD@@iinput \let\@include\FD@@include}
```

TODO `\usepackage`, `\documentclass`?

2.7 Leaving the Package File

```
\endinput
```

2.8 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
v0.21  2012/10/19   \fd@datekind\@tempb -> \fd@refdate,
                    \fd@refdate with \DatesDiffNotices! (bug)
                    \@tempb -> \fd@therefdate; doc. mod.s
v0.3   2012/10/24   \MessageBreak fix, \DatesDiffWarnings
       2012/10/25   doc.: add <date> in sec:readprov,
                    rm. remark on \fd@datesequal,
                    mod. text on \DatesDiffWarnings,
                    -- -> --- before ‘a script might’
v0.4   2012/11/10   \fd@ -> \FD@, doc. v0.7 -> v0.3, v0.21,
                    \filbreak; code for \...AutoChecks
       2012/11/11   doc. another \filbreak, documenenting
                    \...AutoChecks, \...AutoChecks*
v0.41  2013/03/24   doc. restructured and extended
       2013/03/25   ... continued; section with \FileDateLevelChecks
                    etc., \@ifstar without braces;
                    doc. automatical ("archaic") -> automatic,
                    reducing numbers of lines
       2013/03/26   \FileDateUserChecks etc., \FD@check@pdf@job
```

3 Use with Present Package Documentation

Above this paragraph, the documentation source '`filedate.tex`' issues

`\input{fdatechk.tex}`

in order to run the following TEX script '`fdatechk.tex`':

```
\ProvidesFile{fdatechk.tex}[2013/03/25 'filedate' checks]
\EqualityMessages
\ReadFileInfos{filedate.RLS,srcfiles}
\DoWithBasesExts{\CheckDateOfPDFmod}{filedate}{sty,tex,RLS}
\CheckDateOfPDFmod{srcfiles.tex}
\DatesDiffWarnings
\CheckDateOfToday{filedate.RLS}
```

(That is done *above* the paragraph to avoid wrong spacing within the paragraph from '`filedate.tex`'.) This way we check whether the "info dates" of the package file '`filedate.sty`', of the documentation source and driver '`filedate.tex`', and of some other related files are the same as their modification dates according to `\pdffilemoddate` (using `pdflatex`). When I added the (original) check 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. `\DoWithBasesExts` is from the filesdo package (commado [10] bundle).

The TEX script '`srcfiles.tex`' that in the first instance generates a release overview additionally inputs '`fdatechk.tex`' (as of 2012-11-06) as well. This way the check is performed even when I rerun the documentation without updating the file list, as well the other way round.

[10] `http://www.ctan.org/pkg/commado`