

# filedate.sty

## Access and Compare L<sup>A</sup>T<sub>E</sub>X File Info and Modification Date\*

Uwe Lück<sup>†</sup>

November 6, 2012

### Abstract

`filedate.sty` provides basic access to the date of a L<sup>A</sup>T<sub>E</sub>X 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/T<sub>E</sub>X) script.

**Related packages::** `filemod`, `getfiledate`, `zwgetfdate`, `fileinfo`

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

## Contents

<b>1</b>	<b>Features and Usage</b>	<b>2</b>
1.1	Installing and Calling . . . . .	2
1.2	Demonstration with a “T <sub>E</sub> X script” Example . . . . .	2
<b>2</b>	<b>Implementation and Single Commands</b>	<b>3</b>
2.1	Package File Header (Legalese) . . . . .	3
2.2	The <code>readprov</code> Package . . . . .	3
2.3	Accessing “Info Date” . . . . .	4
2.4	Accessing <code>\pdffilemoddate</code> . . . . .	4

---

\*This document describes version [v0.3](#) of `filedate.sty` as of 2012/10/25.

<sup>†</sup><http://contact-ednotes.sty.de.vu>

1	FEATURES AND USAGE	2
2.5	<code>\rawtoday</code>	5
2.6	Comparing	5
2.7	Reporting Styles	6
2.8	Reference Date “Types”	7
2.9	Leaving the Package File	7
2.10	VERSION HISTORY	8
3	Use with Present Package Documentation	8

# 1 Features and Usage

## 1.1 Installing and Calling

The file `filedate.sty` is provided ready, installation only requires putting it somewhere where  $\text{\TeX}$  finds it (which may need updating the filename data base).<sup>1</sup>

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

```
\usepackage{filedate}
```

but in “ $\text{\TeX}$  scripts” such as below,

```
\RequirePackage{filedate}
```

is better.

## 1.2 Demonstration with a “ $\text{\TeX}$ script” Example

The accompanying `wrong.tex` is an example of a “`filedate`  $\text{\TeX}$  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.

---

<sup>1</sup><http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf>

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 L<sup>A</sup>T<sub>E</sub>X’s `\stop` command, without typesetting anything. T<sub>E</sub>X 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}[2012/10/25 v0.3 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>}{<date>}`

(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:

```

20 \newcommand*\theinfodateof[1]{%
21     \ifundefined{ver@#1}{unknown}{%
22         \expandafter\expandafter\expandafter
23             \fd@firstword\csname ver@#1\endcsname@gobble{} \@nil}}
24 \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.

```

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

```

`\ReadInfoDateOf{<file>}` additionally inputs the info before:

```

27 \newcommand*\ReadInfoDateOf[1]{%
28     \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 `XYLaTeX`) 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:

```

29 \ifx\pdffilemoddate\undefined
30     \newcommand*\thepdfmoddateof{%
31         \string\pdffilemoddate\space unavailable.}
32 \else
33     \newcommand*\thepdfmoddateof[1]{%
34         \expandafter \fd@pdftexdate \pdffilemoddate{#1}\@nil}
35     \expandafter \def \expandafter
36         \fd@pdftexdate\string D:#1#2#3#4#5#6#7#8#9\@nil{%
37         #1#2#3#4/#5#6/#7#8}

```

—cf. Will Robertson’s suggestion dating from 2010 on [stackoverflow.com](https://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.

```
38 \fi
```

## 2.5 \rawtoday

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

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

## 2.6 Comparing

`\CheckDateOf{<file>}{<date>}` compares `<file>`’s info date with `<date>`:

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

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

```
42 % \ReadFileInfos{#1}%
```

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

```
43 \LoadInfoDateOf{#1}%
44 % \show\theinfodate
45 \ReadPDFmodDateOf{#1}%
46 \edef\fd@therefdate{#2}%
47 % \show\fd@therefdate
48 \ifx\theinfodate\fd@therefdate
49   \fd@datesequal{#1}%
50 \else
51   \fd@datesdiff{#1}%
52 \fi}
```

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

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

**TODO** provide automatically.

## 2.7 Reporting Styles

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

```
55 \let\fd@dateequal\@gobble
```

`\EqualityMessages` changes this to screen and log messages:

```
56 \newcommand*{\EqualityMessages}{\let\fd@dateequal\fd@equalmess}
57 \def\fd@equalmess#1{\message{ + #1 passed date check + }}
58 \def\fd@errdatesdiff#1{%
59     \PackageError{filedate}{%
60         \fd@infodate{#1}\fd@refdate}{%      %% \fd@refdate 2012/10/19
61         Fix that!}}
```

`\fd@infodate{<file>}` might be used to change the current presentation of the “info date:”

```
62 \def\fd@infodate#1{%
63     #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 reported “drastically” by `\PackageError`:

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

This is the default:

```
65 \DatesDiffErrors
```

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

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

v0.7 adds `\DatesDiffWarnings` to get more salient reports of date differences by `\PackageWarningNoLine`:

```
70 \newcommand*{\DatesDiffWarnings}{\let\fd@datesdiff\fd@warndatesdiff}
71 \def\fd@warndatesdiff#1{%
72     \PackageWarningNoLine{filedate}%
73     {\fd@infodate{#1}\fd@refdate}}
```

After `\ModDates`, reference dates are called “modification” dates:

```
74 \newcommand*{\ModDates}{\let\fd@refdate\fd@moddate}
75 \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`.

```
76 \newcommand*{\SomeDates}{\let\fd@refdate\fd@somedate}
77 \def\fd@somedate{\MessageBreak vs. \fd@therefdate}
```

That’s the default:

```
78 \SomeDates
```

## 2.8 Reference Date “Types”

`\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 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}
```

`\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”.)

```
88 \newcommand*{\CheckDateOfToday}[1]{%
89   \begingroup
90     \def\fd@refdate{%
91       \MessageBreak which is not today}%
92     \CheckDateOf{#1}{\rawtoday}%
93   \endgroup}
```

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

## 2.9 Leaving the Package File

```
94 \endinput
```

## 2.10 VERSION HISTORY

```

95  v0.1    2012/10/15   core try, bad
96  v0.2    2012/10/16   code for first release
97                2012/10/17   reordering, correcting documentation
98  v0.21   2012/10/19   \fd@datekind\@tempb -> \fd@refdate,
99                        \fd@refdate with \DatesDiffNotices! (bug)
100                \@tempb -> \fd@therefdate; doc. mod.s
101  v0.3    2012/10/24   \MessageBreak fix, \DatesDiffWarnings
102                2012/10/25   doc.: add <date> in sec:readprov,
103                        rm. remark on \fd@datesequal,
104                        mod. text on \DatesDiffWarnings,
105                        -- -> --- before 'a script might'
106

```

## 3 Use with Present Package Documentation

Above this paragraph, the documentation source ‘filedate.tex’ issues

```
\input{fdatechk.tex}
```

in order to run the following T<sub>E</sub>X script ‘fdatechk.tex’:

---

```

\ProvidesFile{fdatechk.tex}[2012/11/06 ‘filedate’ checks]
%% TODO short notation
\EqualityMessages
\CheckDateOfPDFmod{filedate.sty}
\CheckDateOfPDFmod{filedate.tex}
\ReadFileInfos{filedate.RLS,srcfiles}
\CheckDateOfPDFmod{filedate.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.

The T<sub>E</sub>X 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.