The rerunfilecheck package

Heiko Oberdiek*
<heiko.oberdiek at googlemail.com>

2016/05/16 v1.8

Abstract

The package provides additional rerun warnings if some auxiliary files have changed. It is based on MD5 checksum, provided by pdfTeX.

Contents

*Please report any issues at https://github.com/ho-tex/oberdiek/issues

1 Documentation

LaTeX informs the user, when to run LaTeX again, if the references have changed. It has the old references from the first reading of the .aux files already in memory, thus it compares them with the new version of the .aux file at the end of the document. However this rerun warnings are not given for the table of contents and other data stored in the .aux files or other auxiliary files. Usually many of these data as the table of contents is not keep in memory. If someone wants to detect changes, he has either to keep the data in memory. This does not scale well with huge documents. Or he copies the file before they are changed. Slow I/O operations cost time.

Since version 1.30.0 pdfTeX provides \pdfmdfivesum and \pdffilesize. These features are also available in LuaTeX, provided by package pdftexcmds. Thus this package rerunfilecheck uses these features to detect file changes. This saves the packages from keeping the whole files in memory or in file copies. The drawback are different files with the same size and the same MD5 checksum (seldom, hopefully).

1.1 Options

All options are key value options of boolean type. No option or true turns an option on, false disables an option.

mainaux: Check the main .aux file.

partaux: Check the .aux files from \include files.

starttoc: Add the rerun checks in \@starttoc that is called by \tableofcontents, \listoffigures, ...

index, glossary: LaTeX's original \makeindex and \makeglossary are redefined to add the rerun checks. The options do not have an effect, if \makeindex/\makeglossary are already called or if a package or class had redefined or will redefine them.

aux: This option turns all previous options on or off. "aux" means auxiliary file.

The default for the options is false, because some internals must be redefined to insert the rerun checks. The options can be set in \usepackage or the configuration file rerunfilecheck.cfg. Global options are ignored (since 1.4).

\RerunFileCheckSetup {⟨*key value list*⟩}

Options can also be set using \RerunFileCheckSetup. Currently all options are disabled after the package is loaded. Thus \RerunFileCheckSetup makes sense in the configuration file only.

Example for the configuration file:

```
1 ⟨*cfg⟩
2 \ProvidesFile{rerunfilecheck.cfg}[2016/05/16 Default configuration]%
3 \RerunFileCheckSetup{aux}
4 ⟨/cfg⟩
```

1.2 Interface for class/package authors

\RerunFileCheck {⟨*file*⟩} {⟨*file closing action*⟩} {⟨*rerun warning*⟩}

If you want to add a rerun check, call \RerunFileCheck right before an output file is opened for writing. The macro first remembers the current checksum of ⟨*file*⟩. The file is checked again right before the end of the job. Macro \AtVeryEndDocument of package atveryend is used to place the check after the main aux file

is closed in \end{document}. Before reading the file again, it must be closed. Provide the code for closing in argument ⟨*file closing action*⟩. Do not forget \immediate before \openout. Otherwise the closing action would be delayed to the next shipout that never happens (the last page is already shipped out). If the file has changed, \RerunFileCheck informs the user with a warning that the file has changed and says the magic word "Rerun". If the last argument ⟨*rerun warning*⟩ is not empty, then the rerun sentence is replaced by it. Usually the phrase "to get something right" is added. As example the relevant part of the redefined \makeindex is shown, see package code:

```
\newwrite\@indexfile
\RerunFileCheck{\jobname.idx}{%
  \immediate\closeout\@indexfile
}{%
  Rerun LaTeX/makeindex to get index right%
}%
\immediate\openout\@indexfile=\jobname.idx %
```

2 Implementation

```
5 ⟨*package⟩
6 \begingroup\catcode61\catcode48\catcode32=10\relax%
7   \catcode13=5 % ^^M
8   \endlinechar=13 %
9   \catcode123=1 % {
10  \catcode125=2 % }
11  \catcode64=11 % @
12  \def\x{\endgroup
13    \expandafter\edef\csname ReFiCh@AtEnd\endcsname{%
14      \endlinechar=\the\endlinechar\relax
15      \catcode13=\the\catcode13\relax
16      \catcode32=\the\catcode32\relax
17      \catcode35=\the\catcode35\relax
18      \catcode61=\the\catcode61\relax
19      \catcode64=\the\catcode64\relax
20      \catcode123=\the\catcode123\relax
21      \catcode125=\the\catcode125\relax
22    }%
23  }%
24 \x\catcode61\catcode48\catcode32=10\relax%
25 \catcode13=5 % ^^M
26 \endlinechar=13 %
27 \catcode35=6 % #
28 \catcode64=11 % @
29 \catcode123=1 % {
30 \catcode125=2 % }
31 \def\TMP@EnsureCode#1#2{%
32   \edef\ReFiCh@AtEnd{%
33     \ReFiCh@AtEnd
34     \catcode#1=\the\catcode#1\relax
35   }%
36   \catcode#1=#2\relax
37 }
38 \TMP@EnsureCode{39}{12}% '
39 \TMP@EnsureCode{40}{12}% (
40 \TMP@EnsureCode{41}{12}% )
41 \TMP@EnsureCode{42}{12}% *
42 \TMP@EnsureCode{44}{12}% ,
43 \TMP@EnsureCode{46}{12}% .
44 \TMP@EnsureCode{47}{12}% /
45 \TMP@EnsureCode{58}{12}% :
```

```
\TMP@EnsureCode{59}{12}% ;
\TMP@EnsureCode{60}{12}% <
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{91}{12}% [
\TMP@EnsureCode{93}{12}% ]
\TMP@EnsureCode{96}{12}% `
\edef\ReFiCh@AtEnd{\ReFiCh@AtEnd\noexpand\endinput}
```

Package identification.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{rerunfilecheck}%
  [2016/05/16 v1.8 Rerun checks for auxiliary files (HO)]
```

2.1 Options

```
\RequirePackage{kvoptions}[2010/02/22]
\SetupKeyvalOptions{%
  family=rerunfilecheck,%
  prefix=ReFiCh@%
}
```

\RerunFileCheckSetup

```
\newcommand*{\RerunFileCheckSetup}{%
  \setkeys{rerunfilecheck}%
}

\DeclareBoolOption{mainaux}
\DeclareBoolOption{partaux}
\DeclareBoolOption{starttoc}
\DeclareBoolOption{index}
\DeclareBoolOption{glossary}
\define@key{rerunfilecheck}{aux}[true]{%
  \RerunFileCheckSetup{%
    mainaux={#1},%
    partaux={#1},%
    starttoc={#1},%
    index={#1},%
    glossary={#1}%
  }%
}

\InputIfFileExists{rerunfilecheck.cfg}{}{}
\ProcessLocalKeyvalOptions*
```

\ReFiCh@DisableOption

```
\def\ReFiCh@DisableOption{%
  \DisableKeyvalOption[%
    action=warning,%
    package=rerunfilecheck%
  ]{rerunfilecheck}%
}
```

2.2 Check for checksum feature

```
\RequirePackage{infwarerr}[2007/09/09]
\RequirePackage{pdftexcmds}[2009/04/10]

\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname pdf@filemdfivesum\endcsname\relax
  \@PackageInfoNoLine{rerunfilecheck}{%
    Feature \string\pdfmdfivesum\space is not available\MessageBreak
    (e.g. pdfTeX or LuaTeX with package `pdftexcmds').\MessageBreak
    Therefore file contents cannot be checked efficiently\MessageBreak
    and the loading of the package is aborted%
  }%
```

```
  \newcommand*{\RerunFileCheck}[3]{}%
  \renewcommand*{\RerunFileCheckSetup}[1]{}%
  \expandafter\ReFiCh@AtEnd
\fi%
```

2.3 Standard .aux files

```
\ifReFiCh@partaux
  \let\ReFiCh@org@include\@include
  \def\@include#1 {%
    \if@filesw
      \RerunFileCheck{#1.aux}{}{}%
    \fi
    \ReFiCh@org@include{#1} %
  }%
\fi
\ifReFiCh@mainaux
  \AtBeginDocument{%
    \ReFiCh@mainauxfalse
  }%
  \ifReFiCh@mainaux
    \AtEndOfPackage{%
      \RerunFileCheck{\jobname.aux}{}{}%
    }%
  \else
    \if@filesw
      \@PackageWarningNoLine{rerunfilecheck}{%
        Main aux file check is disabled,\MessageBreak
        because the file is already opened.\MessageBreak
        Load the package before \string\begin{document}%
      }%
    \fi
  \fi
\fi
\ifReFiCh@starttoc
  \let\ReFiCh@org@starttoc\@starttoc
  \def\@starttoc#1{%
    \if@filesw
      \RerunFileCheck{\jobname.#1}{%
        \@ifundefined{tf@#1}{%
        }{%
          \immediate\closeout\csname tf@#1\endcsname
        }%
      }{}%
    \fi
    \ReFiCh@org@starttoc{#1}%
  }%
\fi
\ifReFiCh@index
  \ifx\makeindex\@empty
    \@PackageWarningNoLine{rerunfilecheck}{%
      Option `index' ignored,\MessageBreak
      because \string\makeindex\space has already been called%
    }%
  \else
    \def\ReFiCh@temp{%
      \newwrite\@indexfile
      \immediate\openout\@indexfile=\jobname.idx %
      \def\index{%
        \@bsphack
        \begingroup
        \@sanitize
        \@wrindex
```

```
    }%
    \typeout{Writing index file \jobname.idx}%
    \let\makeindex\@empty
  }%
  \ifx\ReFiCh@temp\makeindex
    \def\makeindex{%
      \newwrite\@indexfile
      \RerunFileCheck{\jobname.idx}{%
        \immediate\closeout\@indexfile
      }{%
        Rerun LaTeX/makeindex to get index right%
      }%
      \immediate\openout\@indexfile=\jobname.idx %
      \def\index{%
        \@bsphack
        \begingroup
        \@sanitize
        \@wrindex
      }%
      \typeout{Writing index file \jobname.idx}%
      \let\makeindex\@empty
    }%
  \else
    \@PackageInfoNoLine{rerunfilecheck}{%
      Option `index': unsupported version of \string\makeindex
    }%
  \fi
 \fi
\fi
\ifReFiCh@glossary
 \ifx\makeglossary\@empty
  \@PackageWarningNoLine{rerunfilecheck}{%
    Option `glossary' ignored,\MessageBreak
    because \string\makeglossary\space has already been called%
  }%
 \else
  \def\ReFiCh@temp{%
    \newwrite\@glossaryfile
    \immediate\openout\@glossaryfile=\jobname.glo %
    \def\glossary{%
      \@bsphack
      \begingroup
      \@sanitize
      \@wrglossary
    }%
    \typeout{Writing glossary file \jobname.glo }%
    \let\makeglossary\@empty
  }%
  \ifx\ReFiCh@temp\makeglossary
    \def\ReFiCh@temp{%
      \newwrite\@glossaryfile
      \RerunFileCheck{\jobname.glo}{%
        \immediate\closeout\@glossaryfile
      }{%
        Rerun LaTeX/makeindex to get glossary right%
      }%
      \immediate\openout\@glossaryfile=\jobname.glo %
      \def\glossary{%
        \@bsphack
        \begingroup
        \@sanitize
        \@wrglossary
```

```
      }%
      \typeout{Writing glossary file \jobname.glo}%
      \let\makeglossary\@empty
    }%
  \else
    \@PackageInfoNoLine{rerunfilecheck}{%
      Option `glossary': unsupported version of \string\makeglossary
    }%
  \fi
 \fi
\fi
\ReFiCh@DisableOption{mainaux}
\ReFiCh@DisableOption{partaux}
\ReFiCh@DisableOption{starttoc}
\ReFiCh@DisableOption{index}
\ReFiCh@DisableOption{glossary}
\ReFiCh@DisableOption{aux}
```

2.4 Rerun check

```
\RequirePackage{atveryend}[2016/05/16]
\RequirePackage{uniquecounter}[2009/12/18]
```

\ReFiCh@CheckSum

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname pdf@filesize\endcsname\relax
 \def\ReFiCh@CheckSum{%
  \pdf@filemdfivesum
 }%
\else
 \def\ReFiCh@CheckSum#1{%
  \pdf@filemdfivesum{#1}%
  \ReFiCh@Separator
  \pdf@filesize{#1}%
 }%
\fi
```

\ReFiCh@NoFile

```
\def\ReFiCh@Separator{;}
```

\ReFiCh@NoFile

```
\def\ReFiCh@NoFile{<no file>}
```

```
\UniqueCounterNew{rerunfilecheck}
```

\RerunFileCheck

```
\newcommand*{\RerunFileCheck}{%
 \UniqueCounterCall{rerunfilecheck}\ReFiCh@RerunFileCheck
}
```

\ReFiCh@RerunFileCheck

```
\def\ReFiCh@RerunFileCheck#1{%
 \expandafter\ReFiCh@@RerunFileCheck\csname ReFiCh@#1\endcsname
}
```

\ReFiCh@Check

```
\def\ReFiCh@Check#1#2#3{%
% \IfFileExists{#3}{%
   #1\edef#2{\ReFiCh@CheckSum{#3}}%
   \ifx#2\ReFiCh@Separator
    #1\let#2\ReFiCh@NoFile
   \fi
% }{%
```

```
%    #1\let#2\ReFiCh@NoFile
% }%
}
```

\ReFiCh@@RerunFileCheck

```
\def\ReFiCh@@RerunFileCheck#1#2#3#4{%
 \ReFiCh@Check\global#1{#2}%
 \AtEndAfterFileList{%
  \begingroup
   #3%
   \ReFiCh@Check{}\x{#2}%
   \ifx#1\x
    \@PackageInfoNoLine{rerunfilecheck}{%
     File `#2' has not changed.\MessageBreak
     Checksum: \x
    }%
   \else
    \ifnum
     \ReFiCh@IsAux#2\relax.aux\relax\@nil
     \ifx#1\ReFiCh@NoFile 1\else 0\fi
     \ifx\x\ReFiCh@AuxEmptyUnix 1%
     \else
      \ifx\x\ReFiCh@AuxEmptyDos 1\fi
     \fi
    =111 %
     \@PackageInfoNoLine{rerunfilecheck}{%
      File `#2' is empty .aux file.\MessageBreak
      Before: #1\MessageBreak
      After: \space\x
     }%
    \else
     \@PackageWarningNoLine{rerunfilecheck}{%
      File `#2' has changed.%
      \ifx\\#4\\%
       \space Rerun%
      \else
       \MessageBreak
       #4%
      \fi
     }%
     \@PackageInfoNoLine{rerunfilecheck}{%
      Checksums for `#2':\MessageBreak
      Before: #1\MessageBreak
      After: \space\x
     }%
    \fi
   \fi
  \endgroup
 }%
}

\def\ReFiCh@IsAux#1.aux\relax#2\@nil{%
 \ifx\hbox#2\hbox
  0%
 \else
  1%
 \fi
}

\def\ReFiCh@AuxEmptyUnix{A94A2480D3289E625EEA47CD1B285758;8}%
\@onelevel@sanitize\ReFiCh@AuxEmptyUnix

\def\ReFiCh@AuxEmptyDos{A62A15ECE803E2EBE94952FCC9933BC0;9}%
\@onelevel@sanitize\ReFiCh@AuxEmptyDos
```

```
\ReFiCh@AtEnd%
⟨/package⟩
```

3 Test

```
⟨*test1⟩
\def\LoadCommand{\RequirePackage{rerunfilecheck}[2016/05/16]}
⟨/test1⟩
```

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\RangeCatcodeCheck#1#2#3{%
  \count@=#1\relax
  \loop
    \ifnum#3=\catcode\count@
    \else
      \errmessage{%
        Character \the\count@\space
        with wrong catcode \the\catcode\count@\space
        instead of \number#3%
      }%
    \fi
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\def\space{ }
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input rerunfilecheck.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\%=14 %
  \LoadCommand
  \RangeCatcodeCheck{0}{36}{15}%
  \RangeCatcodeCheck{37}{37}{14}%
  \RangeCatcodeCheck{38}{47}{15}%
  \RangeCatcodeCheck{48}{57}{12}%
  \RangeCatcodeCheck{58}{63}{15}%
  \RangeCatcodeCheck{64}{64}{12}%
  \RangeCatcodeCheck{65}{90}{11}%
  \RangeCatcodeCheck{91}{91}{15}%
  \RangeCatcodeCheck{92}{92}{0}%
  \RangeCatcodeCheck{93}{96}{15}%
  \RangeCatcodeCheck{97}{122}{11}%
  \RangeCatcodeCheck{123}{255}{15}%
  \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/rerunfilecheck.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/rerunfilecheck.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 docu-

[1] http://ctan.org/pkg/rerunfilecheck

mentation 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 rerunfilecheck.dtx
```

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

```
rerunfilecheck.sty              → tex/latex/oberdiek/rerunfilecheck.sty
rerunfilecheck.pdf              → doc/latex/oberdiek/rerunfilecheck.pdf
rerunfilecheck-example.cfg      → doc/latex/oberdiek/rerunfilecheck-example.cfg
test/rerunfilecheck-test1.tex → doc/latex/oberdiek/test/rerunfilecheck-test1.tex
rerunfilecheck.dtx              → source/latex/oberdiek/rerunfilecheck.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

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{rerunfilecheck.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 rerunfilecheck.dtx
makeindex -s gind.ist rerunfilecheck.idx
pdflatex rerunfilecheck.dtx
makeindex -s gind.ist rerunfilecheck.idx
pdflatex rerunfilecheck.dtx
```

5 Catalogue

The following XML file can be used as source for the TeX Catalogue. The elements caption and description are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is rerunfilecheck.xml.

```
⟨*catalogue⟩
<?xml version='1.0' encoding='us-ascii'?>
<!DOCTYPE entry SYSTEM 'catalogue.dtd'>
<entry datestamp='$Date$' modifier='$Author$' id='rerunfilecheck'>
  <name>rerunfilecheck</name>
  <caption>Checksum based rerun checks on auxiliary files.</caption>
  <authorref id='auth:oberdiek'/>
  <copyright owner='Heiko Oberdiek' year='2009-2011'/>
  <license type='lppl1.3'/>
  <version number='1.8'/>
  <description>
    The package provides additional rerun warnings if some
    auxiliary files have changed. It is based on MD5 checksum,
    provided by pdfTeX.
    <p/>
    The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
  </description>
  <documentation details='Package documentation'
      href='ctan:/macros/latex/contrib/oberdiek/rerunfilecheck.pdf'/>
  <ctan file='true' path='/macros/latex/contrib/oberdiek/rerunfilecheck.dtx'/>
  <miktex location='oberdiek'/>
  <texlive location='oberdiek'/>
  <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'/>
</entry>
⟨/catalogue⟩
```

6 History

[2009/12/10 v1.0]

- The first version.

[2009/12/12 v1.1]

- Short info shortened.

[2009/12/18 v1.2]

- Required date for package uniquecounter updated because of bug in this package.

[2010/01/25 v1.3]

- Moved from TDS:*/generic/* to TDS:*/latex/*.

[2010/02/22 v1.4]

- The options of this package are recognized only if they are package options. Global options are ignored. This avoids name clashes with class and other package options (for example, class option 'index=totoc').

[2010/03/15 v1.5]

- Call of \pdffivesum is wrapped in \IfFileExists to avoid calls of mktextex if this feature is enabled. However \IfFileExists has file name limitations.

[2010/03/16 v1.6]

- Reverted to version 1.4 and \IfFileExists wrapper of version 1.5 is removed.

[2011/04/15 v1.7]

- Using \AtEndAfterFileList of package atveryend 2011/04/15 v1.6 instead of \AtVeryEndDocument.

[2016/05/16 v1.8]

- Documentation updates.

7 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; plain numbers refer to the code lines where the entry is used.