

# The rerunfilecheck package

Heiko Oberdiek\*  
<heiko.oberdiek at gmail.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

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Options	2
1.2	Interface for class/package authors	2
<b>2</b>	<b>Implementation</b>	<b>3</b>
2.1	Options	4
2.2	Check for checksum feature	4
2.3	Standard .aux files	5
2.4	Rerun check	7
<b>3</b>	<b>Test</b>	<b>9</b>
3.1	Catcode checks for loading	9
<b>4</b>	<b>Installation</b>	<b>10</b>
4.1	Download	10
4.2	Bundle installation	11
4.3	Package installation	11
4.4	Refresh file name databases	11
4.5	Some details for the interested	11
<b>5</b>	<b>Catalogue</b>	<b>12</b>
<b>6</b>	<b>History</b>	<b>12</b>
	[2009/12/10 v1.0]	12
	[2009/12/12 v1.1]	12
	[2009/12/18 v1.2]	12
	[2010/01/25 v1.3]	13
	[2010/02/22 v1.4]	13
	[2010/03/15 v1.5]	13
	[2010/03/16 v1.6]	13
	[2011/04/15 v1.7]	13
	[2016/05/16 v1.8]	13

---

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

# 1 Documentation

L<sup>A</sup>T<sub>E</sub>X informs the user, when to run L<sup>A</sup>T<sub>E</sub>X 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 pdfT<sub>E</sub>X provides `\pdfmdfivesum` and `\pdffilesize`. These features are also available in LuaT<sub>E</sub>X, 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:** L<sup>A</sup>T<sub>E</sub>X'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}% :
```

```

46 \TMP@EnsureCode{59}{12}% ;
47 \TMP@EnsureCode{60}{12}% <
48 \TMP@EnsureCode{62}{12}% >
49 \TMP@EnsureCode{91}{12}% [
50 \TMP@EnsureCode{93}{12}% ]
51 \TMP@EnsureCode{96}{12}% `
52 \edef\ReFiCh@AtEnd{\ReFiCh@AtEnd\noexpand\endinput}

```

Package identification.

```

53 \NeedsTeXFormat{LaTeX2e}
54 \ProvidesPackage{rerunfilecheck}%
55 [2016/05/16 v1.8 Rerun checks for auxiliary files (HO)]

```

## 2.1 Options

```

56 \RequirePackage{kvoptions}[2010/02/22]
57 \SetupKeyvalOptions{%
58   family=rerunfilecheck,%
59   prefix=ReFiCh@%
60 }

```

\RerunFileCheckSetup

```

61 \newcommand*\RerunFileCheckSetup{%
62   \setkeys{rerunfilecheck}%
63 }

64 \DeclareBoolOption{mainaux}
65 \DeclareBoolOption{partaux}
66 \DeclareBoolOption{starttoc}
67 \DeclareBoolOption{index}
68 \DeclareBoolOption{glossary}
69 \define@key{rerunfilecheck}{aux}[true]{%
70   \RerunFileCheckSetup{%
71     mainaux={#1},%
72     partaux={#1},%
73     starttoc={#1},%
74     index={#1},%
75     glossary={#1}%
76   }%
77 }

78 \InputIfFileExists{rerunfilecheck.cfg}{-}{}
79 \ProcessLocalKeyvalOptions*

```

\ReFiCh@DisableOption

```

80 \def\ReFiCh@DisableOption{%
81   \DisableKeyvalOption[%
82     action=warning,%
83     package=rerunfilecheck%
84   ]{rerunfilecheck}%
85 }

```

## 2.2 Check for checksum feature

```

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

88 \begingroup\expandafter\expandafter\expandafter\endgroup
89 \expandafter\ifx\csname pdf@filemdfivesum\endcsname\relax
90   \@PackageInfoNoLine{rerunfilecheck}{%
91     Feature \string\pdfmdfivesum\space is not available\MessageBreak
92     (e.g. pdfTeX or LuaTeX with package `pdftexcmds').\MessageBreak
93     Therefore file contents cannot be checked efficiently\MessageBreak
94     and the loading of the package is aborted%
95   }%

```

```

96 \newcommand*{\RerunFileCheck}[3]{}%
97 \renewcommand*{\RerunFileCheckSetup}[1]{}%
98 \expandafter\ReFiCh@AtEnd
99 \fi%

```

## 2.3 Standard .aux files

```

100 \ifReFiCh@partaux
101 \let\ReFiCh@org@include\@include
102 \def\@include#1 {%
103   \if@filesw
104     \RerunFileCheck{#1.aux}{-}{-}%
105   \fi
106   \ReFiCh@org@include{#1} %
107 }%
108 \fi
109 \ifReFiCh@mainaux
110 \AtBeginDocument{%
111   \ReFiCh@mainauxfalse
112 }%
113 \ifReFiCh@mainaux
114 \AtEndOfPackage{%
115   \RerunFileCheck{\jobname.aux}{-}{-}%
116 }%
117 \else
118   \if@filesw
119     \@PackageWarningNoLine{rerunfilecheck}{%
120       Main aux file check is disabled,\MessageBreak
121       because the file is already opened.\MessageBreak
122       Load the package before \string\begin{document}%
123     }%
124   \fi
125 \fi
126 \fi
127 \ifReFiCh@starttoc
128 \let\ReFiCh@org@starttoc\@starttoc
129 \def\@starttoc#1{%
130   \if@filesw
131     \RerunFileCheck{\jobname.#1}{-}%
132     \@ifundefined{tf@#1}{%
133       }{%
134         \immediate\closeout\csname tf@#1\endcsname
135       }%
136     }{-}%
137   \fi
138   \ReFiCh@org@starttoc{#1}%
139 }%
140 \fi
141 \ifReFiCh@index
142 \ifx\makeindex\@empty
143   \@PackageWarningNoLine{rerunfilecheck}{%
144     Option `index' ignored,\MessageBreak
145     because \string\makeindex\space has already been called%
146   }%
147 \else
148   \def\ReFiCh@temp{%
149     \newwrite\@indexfile
150     \immediate\openout\@indexfile=\jobname.idx %
151     \def\index{%
152       \@bsphack
153       \begingroup
154       \@sanitize
155       \@wrindex

```

```

156 }%
157 \typeout{Writing index file \jobname.idx}%
158 \let\makeindex\@empty
159 }%
160 \ifx\ReFiCh@temp\makeindex
161 \def\makeindex{%
162 \newwrite\@indexfile
163 \RerunFileCheck{\jobname.idx}{%
164 \immediate\closeout\@indexfile
165 }{%
166 Rerun LaTeX/makeindex to get index right%
167 }%
168 \immediate\openout\@indexfile=\jobname.idx %
169 \def\index{%
170 \@bsphack
171 \begingroup
172 \@sanitize
173 \@wrindex
174 }%
175 \typeout{Writing index file \jobname.idx}%
176 \let\makeindex\@empty
177 }%
178 \else
179 \@PackageInfoNoLine{rerunfilecheck}{%
180 Option `index': unsupported version of \string\makeindex
181 }%
182 \fi
183 \fi
184 \fi
185 \ifReFiCh@glossary
186 \ifx\makeglossary\@empty
187 \@PackageWarningNoLine{rerunfilecheck}{%
188 Option `glossary' ignored,\MessageBreak
189 because \string\makeglossary\space has already been called%
190 }%
191 \else
192 \def\ReFiCh@temp{%
193 \newwrite\@glossaryfile
194 \immediate\openout\@glossaryfile=\jobname.glo %
195 \def\glossary{%
196 \@bsphack
197 \begingroup
198 \@sanitize
199 \@wrglossary
200 }%
201 \typeout{Writing glossary file \jobname.glo }%
202 \let\makeglossary\@empty
203 }%
204 \ifx\ReFiCh@temp\makeglossary
205 \def\ReFiCh@temp{%
206 \newwrite\@glossaryfile
207 \RerunFileCheck{\jobname.glo}{%
208 \immediate\closeout\@glossaryfile
209 }{%
210 Rerun LaTeX/makeindex to get glossary right%
211 }%
212 \immediate\openout\@glossaryfile=\jobname.glo %
213 \def\glossary{%
214 \@bsphack
215 \begingroup
216 \@sanitize
217 \@wrglossary

```

```

218     }%
219     \typeout{Writing glossary file \jobname.glo}%
220     \let\makeglossary\@empty
221     }%
222   \else
223     \@PackageInfoNoLine{rerunfilecheck}{%
224       Option `glossary': unsupported version of \string\makeglossary
225     }%
226   \fi
227 \fi
228 \fi
229 \ReFiCh@DisableOption{mainaux}
230 \ReFiCh@DisableOption{partaux}
231 \ReFiCh@DisableOption{starttoc}
232 \ReFiCh@DisableOption{index}
233 \ReFiCh@DisableOption{glossary}
234 \ReFiCh@DisableOption{aux}

```

## 2.4 Rerun check

```

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

```

\ReFiCh@Checksum

```

237 \begingroup\expandafter\expandafter\expandafter\endgroup
238 \expandafter\ifx\csname pdf@filesize\endcsname\relax
239   \def\ReFiCh@Checksum{%
240     \pdf@filemdfivesum
241   }%
242 \else
243   \def\ReFiCh@Checksum#1{%
244     \pdf@filemdfivesum{#1}%
245     \ReFiCh@Separator
246     \pdf@filesize{#1}%
247   }%
248 \fi

```

\ReFiCh@NoFile

```

249 \def\ReFiCh@Separator{;}

```

\ReFiCh@NoFile

```

250 \def\ReFiCh@NoFile{<no file>}
251 \UniqueCounterNew{rerunfilecheck}

```

\RerunFileCheck

```

252 \newcommand*{\RerunFileCheck}{%
253   \UniqueCounterCall{rerunfilecheck}\ReFiCh@RerunFileCheck
254 }

```

\ReFiCh@RerunFileCheck

```

255 \def\ReFiCh@RerunFileCheck#1{%
256   \expandafter\ReFiCh@@RerunFileCheck\csname ReFiCh@#1\endcsname
257 }

```

\ReFiCh@Check

```

258 \def\ReFiCh@Check#1#2#3{%
259 %   \IfFileExists{#3}{%
260   #1\edef#2{\ReFiCh@Checksum{#3}}%
261   \ifx#2\ReFiCh@Separator
262     #1\let#2\ReFiCh@NoFile
263   \fi
264 %   }{%

```

```

265 %   #1\let#2\ReFiCh@NoFile
266 % }%
267 }

```

\ReFiCh@@RerunFileCheck

```

268 \def\ReFiCh@@RerunFileCheck#1#2#3#4{%
269   \ReFiCh@Check\global#1{#2}%
270   \AtEndAfterFileList{%
271     \begingroup
272     #3%
273     \ReFiCh@Check{}\x{#2}%
274     \ifx#1\x
275       \@PackageInfoNoLine{rerunfilecheck}{%
276         File `#2' has not changed.\MessageBreak
277         Checksum: \x
278       }%
279     \else
280       \ifnum
281         \ReFiCh@IsAux#2\relax.aux\relax\@nil
282         \ifx#1\ReFiCh@NoFile 1\else 0\fi
283         \ifx\x\ReFiCh@AuxEmptyUnix 1%
284         \else
285           \ifx\x\ReFiCh@AuxEmptyDos 1\fi
286         \fi
287       =111 %
288       \@PackageInfoNoLine{rerunfilecheck}{%
289         File `#2' is empty .aux file.\MessageBreak
290         Before: #1\MessageBreak
291         After: \space\x
292       }%
293     \else
294       \@PackageWarningNoLine{rerunfilecheck}{%
295         File `#2' has changed.%
296         \ifx\#4\%
297           \space Rerun%
298         \else
299           \MessageBreak
300           #4%
301         \fi
302       }%
303       \@PackageInfoNoLine{rerunfilecheck}{%
304         Checksums for `#2':\MessageBreak
305         Before: #1\MessageBreak
306         After: \space\x
307       }%
308     \fi
309   \fi
310   \endgroup
311 }%
312 }

313 \def\ReFiCh@IsAux#1.aux\relax#2\@nil{%
314   \ifx\hbox#2\hbox
315     0%
316   \else
317     1%
318   \fi
319 }

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

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

```



```

324 \ReFiCh@AtEnd%
325 \</package>

```

### 3 Test

```

326 \*test1\
327 \def\LoadCommand{\RequirePackage{rerunfilecheck}[2016/05/16]}
328 \</test1\

```

#### 3.1 Catcode checks for loading

```

329 \*test1\
330 \catcode`\{=1 %
331 \catcode`\}=2 %
332 \catcode`\#=6 %
333 \catcode`\@=11 %
334 \expandafter\ifx\csname count@\endcsname\relax
335   \countdef\count@=255 %
336 \fi
337 \expandafter\ifx\csname @gobble\endcsname\relax
338   \long\def\@gobble#1{}%
339 \fi
340 \expandafter\ifx\csname @firstofone\endcsname\relax
341   \long\def\@firstofone#1{#1}%
342 \fi
343 \expandafter\ifx\csname loop\endcsname\relax
344   \expandafter\@firstofone
345 \else
346   \expandafter\@gobble
347 \fi
348 {%
349   \def\loop#1\repeat{%
350     \def\body{#1}%
351     \iterate
352   }%
353   \def\iterate{%
354     \body
355     \let\next\iterate
356   \else
357     \let\next\relax
358   \fi
359   \next
360 }%
361 \let\repeat=\fi
362 }%
363 \def\RestoreCatcodes{}
364 \count@=0 %
365 \loop
366   \edef\RestoreCatcodes{%
367     \RestoreCatcodes
368     \catcode\the\count@=\the\catcode\count@\relax
369   }%
370 \ifnum\count@<255 %
371   \advance\count@ 1 %
372 \repeat
373
374 \def\RangeCatcodeInvalid#1#2{%
375   \count@=#1\relax
376   \loop
377     \catcode\count@=15 %
378     \ifnum\count@<#2\relax
379       \advance\count@ 1 %

```

```

380 \repeat
381 }
382 \def\RangeCatcodeCheck#1#2#3{%
383 \count@=#1\relax
384 \loop
385 \ifnum#3=\catcode\count@
386 \else
387 \errmessage{%
388 Character \the\count@\space
389 with wrong catcode \the\catcode\count@\space
390 instead of \number#3%
391 }%
392 \fi
393 \ifnum\count@<#2\relax
394 \advance\count@ 1 %
395 \repeat
396 }
397 \def\space{ }
398 \expandafter\ifx\csname LoadCommand\endcsname\relax
399 \def\LoadCommand{\input rerunfilecheck.sty\relax}%
400 \fi
401 \def\Test{%
402 \RangeCatcodeInvalid{0}{47}%
403 \RangeCatcodeInvalid{58}{64}%
404 \RangeCatcodeInvalid{91}{96}%
405 \RangeCatcodeInvalid{123}{255}%
406 \catcode`\@=12 %
407 \catcode`\=0 %
408 \catcode`\%=14 %
409 \LoadCommand
410 \RangeCatcodeCheck{0}{36}{15}%
411 \RangeCatcodeCheck{37}{37}{14}%
412 \RangeCatcodeCheck{38}{47}{15}%
413 \RangeCatcodeCheck{48}{57}{12}%
414 \RangeCatcodeCheck{58}{63}{15}%
415 \RangeCatcodeCheck{64}{64}{12}%
416 \RangeCatcodeCheck{65}{90}{11}%
417 \RangeCatcodeCheck{91}{91}{15}%
418 \RangeCatcodeCheck{92}{92}{0}%
419 \RangeCatcodeCheck{93}{96}{15}%
420 \RangeCatcodeCheck{97}{122}{11}%
421 \RangeCatcodeCheck{123}{255}{15}%
422 \RestoreCatcodes
423 }
424 \Test
425 \csname @@end\endcsname
426 \end
427 </test1>

```

## 4 Installation

### 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

[CTAN:macros/latex/contrib/oberdiek/rerunfilecheck.dtx](http://ctan.org/macros/latex/contrib/oberdiek/rerunfilecheck.dtx) The source file.

[CTAN:macros/latex/contrib/oberdiek/rerunfilecheck.pdf](http://ctan.org/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-

<sup>1</sup><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 T<sub>E</sub>X 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 T<sub>E</sub>X:

```
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):

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

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 T<sub>E</sub>X distribution (teT<sub>E</sub>X, mikT<sub>E</sub>X, ...) relies on file name databases, you must refresh these. For example, teT<sub>E</sub>X users run **texhash** or **mktextlsr**.

## 4.5 Some details for the interested

**Unpacking with L<sup>A</sup>T<sub>E</sub>X.** The **.dtx** chooses its action depending on the format:

**plain T<sub>E</sub>X:** Run **docstrip** and extract the files.

**L<sup>A</sup>T<sub>E</sub>X:** Generate the documentation.

If you insist on using L<sup>A</sup>T<sub>E</sub>X for **docstrip** (really, **docstrip** does not need L<sup>A</sup>T<sub>E</sub>X), 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 pdfL<sup>A</sup>T<sub>E</sub>X:

```
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 [T<sub>E</sub>X 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`.

```
428 (*catalogue)
429 <?xml version='1.0' encoding='us-ascii'?>
430 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
431 <entry datestamp='$Date$' modifier='$Author$' id='rerunfilecheck'>
432   <name>rerunfilecheck</name>
433   <caption>Checksum based rerun checks on auxiliary files.</caption>
434   <authorref id='auth:oberdiek'>
435     <copyright owner='Heiko Oberdiek' year='2009-2011'>
436       <license type='lppl1.3'>
437         <version number='1.8'>
438           <description>
439             The package provides additional rerun warnings if some
440             auxiliary files have changed. It is based on MD5 checksum,
441             provided by pdfTeX.
442           <p/>
443             The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
444           </description>
445           <documentation details='Package documentation'
446             href='ctan:/macros/latex/contrib/oberdiek/rerunfilecheck.pdf'>
447             <ctan file='true' path='/macros/latex/contrib/oberdiek/rerunfilecheck.dtx'>
448             <miktex location='oberdiek'>
449             <texlive location='oberdiek'>
450             <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'>
451           </entry>
452 </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 `\pdfvivesum` 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.

Symbols	A
<code>\#</code> . . . . .	332
<code>\%</code> . . . . .	408
<code>\@</code> . . . . .	333, 406
<code>\@PackageInfoNoLine</code> . . . . .	
. . . . .	90, 179, 223, 275, 288, 303
<code>\@PackageWarningNoLine</code> . . . . .	
. . . . .	119, 143, 187, 294
<code>\@bsphack</code> . . . . .	152, 170, 196, 214
<code>\@empty</code> . . . . .	142, 158, 176, 186, 202, 220
<code>\@firstofone</code> . . . . .	341, 344
<code>\@glossaryfile</code> . . . . .	193, 194, 206, 208, 212
<code>\@gobble</code> . . . . .	338, 346
<code>\@ifundefined</code> . . . . .	132
<code>\@include</code> . . . . .	101, 102
<code>\@indexfile</code> . . . . .	149, 150, 162, 164, 168
<code>\@nil</code> . . . . .	281, 313
<code>\@onelevel@sanitize</code> . . . . .	321, 323
<code>\@sanitize</code> . . . . .	154, 172, 198, 216
<code>\@starttoc</code> . . . . .	128, 129
<code>\@wrglossary</code> . . . . .	199, 217
<code>\@wrindex</code> . . . . .	155, 173
<code>\@</code> . . . . .	296, 407
<code>\{</code> . . . . .	330
<code>\}</code> . . . . .	331
<code>\advance</code> . . . . .	371, 379, 394
<code>\AtBeginDocument</code> . . . . .	110
<code>\AtEndAfterFileList</code> . . . . .	270
<code>\AtEndOfPackage</code> . . . . .	114
B	
<code>\begin</code> . . . . .	122
<code>\body</code> . . . . .	350, 354
C	
<code>\catcode</code> 6, 7, 9, 10, 11, 15, 16, 17, 18, 19, 20, 21, 24, 25, 27, 28, 29, 30, 34, 36, 330, 331, 332, 333, 368, 377, 385, 389, 406, 407, 408	
<code>\closeout</code> . . . . .	134, 164, 208
<code>\count@</code> . . . . .	335, 364, 368, 370, 371, 375, 377, 378, 379, 383, 385, 388, 389, 393, 394
<code>\countdef</code> . . . . .	335
<code>\csname</code> . . . . .	13, 89, 134, 238, 256, 334, 337, 340, 343, 398, 425
D	
<code>\DeclareBoolOption</code> . . . . .	64, 65, 66, 67, 68

<code>\define@key</code> . . . . .	69		<b>P</b>	
<code>\DisableKeyvalOption</code> . . . . .	81		<code>\pdf@filemdfivesum</code> . . . . .	240, 244
<b>E</b>			<code>\pdf@filesize</code> . . . . .	246
<code>\end</code> . . . . .	426		<code>\pdfmdfivesum</code> . . . . .	91
<code>\endcsname</code> . . . . .	13, 89, 134, 238, 256, 334, 337, 340, 343, 398, 425		<code>\ProcessLocalKeyvalOptions</code> . . . . .	79
<code>\endinginput</code> . . . . .	52		<code>\ProvidesFile</code> . . . . .	2
<code>\endlinechar</code> . . . . .	8, 14, 26		<code>\ProvidesPackage</code> . . . . .	54
<code>\errmessage</code> . . . . .	387		<b>R</b>	
<b>G</b>			<code>\RangeCatcodeCheck</code> . . . . .	
<code>\glossary</code> . . . . .	195, 213		. . . . .	382, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421
<b>H</b>			<code>\RangeCatcodeInvalid</code> . . . . .	
<code>\hbox</code> . . . . .	314		. . . . .	374, 402, 403, 404, 405
<b>I</b>			<code>\ReFiCh@@RerunFileCheck</code> . . . . .	256, 268
<code>\if@filesw</code> . . . . .	103, 118, 130		<code>\ReFiCh@AtEnd</code> . . . . .	32, 33, 52, 98, 324
<code>\IfFileExists</code> . . . . .	259		<code>\ReFiCh@AuxEmptyDos</code> . . . . .	285, 322, 323
<code>\ifnum</code> . . . . .	280, 370, 378, 385, 393		<code>\ReFiCh@AuxEmptyUnix</code> . . . . .	283, 320, 321
<code>\ifReFiCh@glossary</code> . . . . .	185		<code>\ReFiCh@Check</code> . . . . .	258, 269, 273
<code>\ifReFiCh@index</code> . . . . .	141		<code>\ReFiCh@Checksum</code> . . . . .	237, 260
<code>\ifReFiCh@mainaux</code> . . . . .	109, 113		<code>\ReFiCh@DisableOption</code> . . . . .	
<code>\ifReFiCh@partaux</code> . . . . .	100		. . . . .	80, 229, 230, 231, 232, 233, 234
<code>\ifReFiCh@starttoc</code> . . . . .	127		<code>\ReFiCh@IsAux</code> . . . . .	281, 313
<code>\ifx</code> . . . . .	89, 142, 160, 186, 204, 238, 261, 274, 282, 283, 285, 296, 314, 334, 337, 340, 343, 398		<code>\ReFiCh@mainauxfalse</code> . . . . .	111
<code>\immediate</code> . . . . .			<code>\ReFiCh@NoFile</code> . . . . .	249, 250, 262, 265, 282
. . . . .	134, 150, 164, 168, 194, 208, 212		<code>\ReFiCh@org@include</code> . . . . .	101, 106
<code>\index</code> . . . . .	151, 169		<code>\ReFiCh@org@starttoc</code> . . . . .	128, 138
<code>\input</code> . . . . .	399		<code>\ReFiCh@RerunFileCheck</code> . . . . .	253, 255
<code>\InputIfFileExists</code> . . . . .	78		<code>\ReFiCh@Separator</code> . . . . .	245, 249, 261
<code>\iterate</code> . . . . .	351, 353, 355		<code>\ReFiCh@temp</code> . . . . .	148, 160, 192, 204, 205
<b>J</b>			<code>\renewcommand</code> . . . . .	97
<code>\jobname</code> . . . . .	115, 131, 150, 157, 163, 168, 175, 194, 201, 207, 212, 219		<code>\repeat</code> . . . . .	349, 361, 372, 380, 395
<b>L</b>			<code>\RequirePackage</code> . . . . .	56, 86, 87, 235, 236, 327
<code>\LoadCommand</code> . . . . .	327, 399, 409		<code>\RerunFileCheck</code> . . . . .	2, 96, 104, 115, 131, 163, 207, 252
<code>\loop</code> . . . . .	349, 365, 376, 384		<code>\RerunFileCheckSetup</code> . . . . .	2, 3, 61, 70, 97
<b>M</b>			<code>\RestoreCatcodes</code> . . . . .	363, 366, 367, 422
<code>\makeglossary</code> . . . . .	186, 189, 202, 204, 220, 224		<b>S</b>	
<code>\makeindex</code> . . . . .			<code>\setkeys</code> . . . . .	62
. . . . .	142, 145, 158, 160, 161, 176, 180		<code>\SetupKeyvalOptions</code> . . . . .	57
<code>\MessageBreak</code> . . . . .			<code>\space</code> . . . . .	91, 145, 189, 291, 297, 306, 388, 389, 397
. . . . .	91, 92, 93, 120, 121, 144, 188, 276, 289, 290, 299, 304, 305		<b>T</b>	
<b>N</b>			<code>\Test</code> . . . . .	401, 424
<code>\NeedsTeXFormat</code> . . . . .	53		<code>\the</code> . . . . .	14, 15, 16, 17, 18, 19, 20, 21, 34, 368, 388, 389
<code>\newcommand</code> . . . . .	61, 96, 252		<code>\TMP@EnsureCode</code> . . . . .	31, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51
<code>\newwrite</code> . . . . .	149, 162, 193, 206		<code>\typeout</code> . . . . .	157, 175, 201, 219
<code>\next</code> . . . . .	355, 357, 359		<b>U</b>	
<code>\number</code> . . . . .	390		<code>\UniqueCounterCall</code> . . . . .	253
<b>O</b>			<code>\UniqueCounterNew</code> . . . . .	251
<code>\openout</code> . . . . .	150, 168, 194, 212		<b>X</b>	
			<code>\x</code> . . . . .	12, 24, 273, 274, 277, 283, 285, 291, 306