pkgcheck Utility, v4.1.0

Manfred Lotz

August 5, 2026

Contents

Abstract

This document describes the pkgcheck command line utility which is used by the author when checking uploaded packages to CTAN.

1 Introduction

Uploaded packages to CTAN must satisfy various requirements in order to get installed on CTAN.

A first introduction is given here https://ctan.org/help/upload-pkg.

Even more details are to be found in the excellent CTAN-upload addendum https://ctan.org/file/help/ctan/CTAN-upload-addendum written by Petra Rübe-Pugliese.

The pkgcheck utility which runs on Linux systems only checks those requirements which can be checked by a program.

2 pkgcheck utility

The pkgcheck utility is a compiled program written in the Rust programming language. It may run in a Linux or macOS environment.

Currently, Windows is not supported. Simply, because the author doesn't use Windows at all.

It will be invoked from the command line, and any error or warning message has a certain message id. pkgcheck offers an option to get more information for a certain error.

3 Requirements

pkgcheck doesn't have any special runtime requirements.

The pkgcheck is a 64-bit statically linked binary, and should work an any 64-bit Linux. It is available in the repository in directory bin/.

Binaries for macOS are also available for aarch64 and x86_64 architecture in the bin/ directory.

4 Installation

Copy the binary from bin/pkgcheck to a suitable location on your hard disk, and (recommended) make sure the directory is in the PATH or call pkgcheck using an absolute path name.

5 Utility usage

5.1 Config file

It is now possible to have a config file for pkgcheck which must be a YAML file. A config file can be specified by using the command line option --config-file. If no config file is specified during invocation of pkgcheck then pkgcheck checks two locations for existence of a config file.

- ~/.ctan/pkgcheck.yml

- `.config/ctan/pkgcheck.yml`

Currently, the config file may contain only TDS path exceptions. For more details see message I0009x.

5.2 Generate shell completions

pkgcheck offers an option to generate shell completions for various shells, most notably bash, zsh and fish.

Run

```
pkgcheck -generate-completion <SHELL>
```

and follow the instructions given.

5.3 Check a package

A package for CTAN is supposed to be uploaded as a ZIP or a g-zipped tar archive. The package must have a top level directory.

After unpacking the archive of a package `mypkg` into directory `mypkg/` it can be checked by running pkgcheck with option `--package-dir` or shorter `-d`.

```
pkgcheck -d mypkg
```

pkgcheck returns 1 if there are any errors, otherwise 0.

5.4 Check a package which has a TDS archive

If a package contains a TDS ZIP archive it is supposed to be in the top level directory of a package.

In order to check the TDS ZIP archive the option `-T <tds_zip>` or `--tds-zip <tds_zip>` can be used.

Please note that a TDS ZIP archive will always be checked together with the non-install tree of the package which means that `--tds-zip` requires option `--package-dir` as well.

Checking package `mypkg` pkgcheck will be invoked like follows:

```
pkgcheck -d mypkg -T mypkg.tds.zip
```

As before pkgcheck returns 1 if there are any errors, otherwise 0.

5.5 Pkgcheck messages

pkgcheck issues three kind of messages

- Information messages
- Warning messages

- Error messages

Messages have unique ids and the detailed explanation of a message can be either looked up in this document, or it can be displayed by using command line option `--explain` or `-e`.

Example

```
pkgcheck --explain e0012
```

5.6 Duplicate files

By default, `pkgcheck` detects duplicate files in a package. This could be disabled by using command line switch `--ignore dupes` or shorter `-I dupes`.

5.7 Same named files

By default, `pkgcheck` detects same named files in a package. This could be disabled by using the command line switch `--ignore same-named` or shorter version `-I same-named`.

5.8 Temporary files

By default, `pkgcheck` detects temporary files in a package. This could be disabled by using the command line switch `--ignore tmpfiles` or the shorter version `-I tmpfiles`

5.9 Permissions

`pkgcheck` offers the option `--correct-le` or shorter `-L` to correct wrong permissions in a package.

5.10 CRLF line endings

`pkgcheck` detects CRLF line endings in text files as good as it can. It reads up to 1 MB to check for CRLF line endings.

Option `--correct-crlf` or for short `-L` can be used to convert a file from CRLF to LF line endings.

5.11 Help options

This section describes specific help related options.

- `-V`
  Outputs `pkgcheck`'s version number.
- `--help`
  Shows the available command line options.
- `--explain <explain`
  This option explains an error message in more detail.
  Example:

```
pkgcheck -e e0012
```

- `--explain-all`
  Outputs a list of explanations of all messages.
- `--show-temp-endings`
  Outputs a list of all file name endings which `pkgcheck` uses to detect temporary files.

6 About checking file types

`pkgcheck` determines, similar to the UNIX `file` command, the type of file. This is required before for example checking permissions or complaining that a text file has CRLF line endings.

It is very important to note that determining file types is not bullet proof. So, it might happen in some cases that `pkgcheck` makes mistakes when determining a file type. This could lead to a subsequent mistake when complaining about an x-bit, or complaining about CRF line ending.

7 About permissions checking

pkgcheck checks

- permission bits in the unpacked package tree
- permission bits in the TDS zip archive (since pkgcheck 4.1.0)
- existence of special bits (setuid/setgit/sticky bit) in both unpacked package tree and TDS zip archive (since pkgcheck 4.1.0). Although there is no danger because special bits get skipped when unpacking a zip or tar archive it is an error from our perspective.

From an installation point of view the files and directories of a package

- must be at least world readable
- must be writable by owner or group
- must not have the x-bit on for the owner if the file isn't an executable, i.e. a script or binary

The reason for this minimal requirement is that the installation utility used by the CTAN team (which by the way was written by Rainer Schöpf a long time ago) sets permissions correctly if the owner permission is set correctly.

Examples:

- `README.md` with 666 is ok because the installation utility converts the permission to 664
- `README.md` with 660 is wrong because the installation utility wouldn't have access to the file

- some.pdf with 744 would be wrong because a PDF document must not have the x-bit on for the owner

Because of the smartness of the installation utility pkgcheck does check minimal requirements only, i.e some weird looking permissions like the 666 above are accepted.

8 Different kind of messages

Innnn Informational messages
: These are message which usually announce pkgcheck actions.

Ennnn Error messages
: Error messages report errors which must be fixed before installing a package.

Wnnnn Warning messages
: Warning messages denote possible errors depending upon the situation.
For example, for a font package having many duplicate files might be ok. For another package it could be regarded as an error.

Fatal messages Fatal messages are related to environmental issues and not related to packages. They are usually unrecoverable errors, and pkgcheck's only option is to terminate. Fatal messages are written to standard error.
If for example, the package directory specified at the command line doesn't exist then the only option is to issue a message and to terminate.

9 Environmental messages

9.1 Error: creating tempdir

There was an error to create a temporary directory for unzipping the TDS zip archive. The error message contains more detailed information what the cause is.

9.2 Error: Config file could not be read

There was an error reading the config file.

9.3 Error: TDS archive is not a zip archive

The TDS archive specified at the command line is not a zip archive.

9.4 Specified TDS archive does not exist or is no file

The TDS archive specified at the command does not exist or is no file.

10 Informational messages

10.1 I0001 – Successfully converted from CRLF to LF

Just an information that pkgcheck has successfully converted a file from CRLF to LF line endings.

10.2 I0002 – Checking package files in directory

Just an information that pkgcheck starts checking the package files in the unzipped directory trees.

10.3 I0003 – Checking TDS zip archive

Just an information that pkgcheck starts checking the TDS zip archive.

10.4 I0004 – Correcting line endings for file

The file had CRLF line ending and will be corrected to have LF (Unix like) line endings.
For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#crlf

10.5 I0005 – Correcting permissions for file or directory

pkgcheck corrects wrong permsissions for package files and directories.
For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#filepermissions

10.6 I0006 – Files having one of the following file name endings are regarded as temporary

Option `--show-temp-endings` was used, and pkgcheck prints a list of temporary file endings and their meanings.
For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#noauxfiles

10.7 I0007 – Successfully corrected wrong line endings to LF resp. CRLF

pkgcheck successfully converted wrong line endings to LF line endings or to CRLF line endings if the file was a Windows text file.
Wrong line endings could be CR, CRLF or a mixture of line endings.

10.8 I0008 – Using config file

Tells the user which config file is used.

10.9 I0009 – Updating entry <pkgname> -> <tpkg> with <pkgname> -> <new tpkg> from config file

This message can only show up if `pkgcheck` got called with `--config`. It is required that the path names in the TDS zip archive contain the package name. There are exceptions however, and if such an exception is defined in the config file `pkgcheck` reports the usage of an exception.

Example: `tex/latex/microtype/microtype-luatex.de` in the TDS zip archive `microtype.tds.zip` contains the package name `microtype`.

There are exceptions, however.

Example: The file names in `latex-amsmath.tds.zip` do not contain `latex-amsmath` but just `latex`.

These exceptions are hard-coded in pkgcheck but can be overridden in the `pkgcheck.yml` config file like in the following example.

```
tds_path_exceptions:
  - pkg: latex-amsmath
    tpkg: latexnew
```

11 Warning messages

11.1 W0001 – Archive as package file detected

Usually a CTAN package should not contain archives. An exception are situations where, for example, the source code of a package is kept in a separate zip archive.

11.2 W0002 – Duplicate files detected

Duplicate files were detected which are listed right after this message.

The message is a warning message as something like this could not be seen as an error in general.

11.3 W0003 – Same named files detected in the package tree

We like to have unique file names over the whole package directory tree. When we discover same named files we report it as a warning. Common names like `README`, `README.txt`, `README.md`, `Makefile`, `Makefile.in`, `Makefile.am` and `makefile` are ignored when checking.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#uniquefilenames

11.4 W0004 – encoding with BOM detected

A UTF encoded package file contains a BOM (byte order mark). Currently, we issues a warning.

Nevertheless, the CTAN team discourages uses of BOM. Please be aware, that in some future time this could be reagarded as an error.

11.5 W0005 – Very large file with size <size> detected in package

(Experimental) We issue the message if there is a file is larger than 40MiB in the package directory tree.

11.6 W0006 – Very large file with size <size> detected in TDS zip archive

(Experimental) We issue the message if there is a file larger than 40MiB in the TDS zip archive.

11.7 W0007 – Empty directory detected in the TDS zip archive

Empty directories in a TDS zip archive are discouraged. As they usually don't create errors in the distribution we issue a warning only.

11.8 W0008 – Windows file has Unix line endings

A Windows file with Unix line endings was detected.

We regard a file as a Windows file if its name ends with:

- `.bat`
- `.cmd`

CRLF for such files is ok but we issue a warning as there may be some kind of edge cases where CRLF is better.

We ignore Powershell files (`.ps1`, `.psm1` or `.psd1`) as both CRLF and LF line endings are ok here.

11.9 W0009 – Replacing <pkgname> → <tpkg> with the same from config file

This message can only show up if `pkgcheck` got called with `--config`. Indicates that an entry in the `pkgcheck` config file does the same as the hard-coded entry. This helps to keep a clean config file.

11.10 W0010 – Hardlinks detected with inode

Hardlinks found in the package directory tree. The inode number will be displayed.

11.11 W0011 – has an mtime in the future by seconds, or <hours, minutes, seconds>

The file `<file>` has a future modification time. This is most probably caused by the archiver tool which doesn't pay attention to the timezone when adding the file to the archive.

The future time will be displayed in

- seconds
- and hours, minutes and seconds

12 Error messages

12.1 E0001 – Bad characters in file name

File name should not contain non-ASCII characters. Additionally, file names should not contain control characters or other characters which may have a special meaning for UNIX shells.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#nounixspecialcharacters

12.2 E0002 – File Permissions

Files submitted to CTAN should be world readable.

Binaries should have the x-bit set.

Scripts with a shebang may have the x-bit set but this is not required.

The same for Windows command files which are files ending in one of `.bat`, `.cmd`, `.ps1`, `.nsh` or `.reg`. x-bit may be set but is not required.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#filepermissions

12.3 E0003 – README is not a text file

The `README` file specified in the error message must be a text file but it isn't.

12.4 E0004 – Empty directory not allowed

Empty directories are considered as rubbish, and are usually not accepted as part of a package, neither in the package tree nor in the TDS zip archive.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#noemptyfiles

12.5 E0005 – Empty files not allowed

Empty files are considered as rubbish, and are usually not accepted as part of a package.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#noemptyfiles

12.6 E0006 – Hidden directories not allowed

A package should not contain hidden directories, neither in the package tree nor in the TDS zip archive.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#noauxfiles

12.7 E0007 – Hidden files not allowed

A package should not contain hidden files, neither in the package tree nor in the TDS zip archive.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#noauxfiles

12.8 E0008 – Temporary file detected

A temporary file was detected. These are typically files created by TeX & friends and should not be part of a package.

Temporary files will also be detected in a TDS zip archive.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#noauxfiles

12.9 E0009 – Package doesn't contain a README file

A package must contain at least one of `README`, `README.md` or `README.txt` file.

For more details refer to:
https://mirror.ctan.org/CTAN/help/ctan/CTAN-upload-addendum.html#readme

12.10 E0010 – Broken symlink detected

A broken symlink was detected.

12.11 E0011 – Wrong permission for directory

Directories should have `rwx` for the owner and at least `r-x` for others (i.e. world readable).

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#filepermissions

12.12 E0012 – CRLF line endings detected

The file specified in the error message contains CRLF line endings. Text files should have UNIX style line endings.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#crlf

12.13 E0013 – Socket special file detected

The file specified in the error message is a socket special file which is not allowed.

12.14 E0014 – Fifo special file detected

The file specified in the error message is a fifo special file which is not allowed.

12.15 E0015 – Block device file detected

The file specified in the error message is a block device file which is not allowed.

12.16 E0016 – Character device file detected

The file specified in the error message is a character device file which is not allowed.

12.17 E0017 – PDF document is in error

The PDF document mentioned in the message is in error.

Example:

```
I0002   Checking package files in directory somepkg
E0017   PDF error detected in somepkg/sompkg.pdf

Error opening "tests/e0017/somepkg/somepkg.pdf": Parse(InvalidTrailer)
```

12.18 E0018 – Unwanted directory detected

A directory was detected which should not be part of a package. Example: `__MACOSX`

12.19 E0019 – Generated file detected

In order to avoid redundancy we don't want to have included files in a package which easily can be generated from other files in the submission.

Exceptions are the `README` files of the package, i.e. `README`, `README.md` or `README.txt`, `.pdf`, `.html`, or `.css` files. If a `.ins` file is a generated file it also will be accepted.

`pkgcheck` detects generated files anywhere in the package directory tree.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#nogeneratedfiles

12.20 E0020 – Unwanted directory detected in the top level directory in TDS zip archive

The name of a top level directory of a TDS archive must be one of those listed here: `asymptote`, `bibtex`, `chktex`, `context`, `doc`, `dvipdfmx`, `dvips`, `fonts`, `hbf2gf`, `makeindex`, `metafont`, `metapost`, `mft`, `omega`, `pbibtex`, `psutils`, `scripts`, `source`, `tex`, `tex4ht`, `texconfig`, `texdoc`, `texdoctk`, `ttf2pk`, `web2c`, `xdvi`, `xindy`,

Any other other directory at the top level is an error.

12.21 E0021 – Error when reading a file

An error was encountered when reading the file specified in the message.

12.22 E0022 – Check of an URL in a README file failed

URL checking is in effect. An error occcurred when trying to retrieve an URL which was found in the specified `README` file.

12.23 E0023 – Follow up error when trying to read a directory with insufficient permissions

Error which is a follow-up error. For instance, when a directory could not be read.

12.24 E0024 – TDS zip archive has wrong permissions

The TDS zip archive should have at least `r--` for the owner and at least `r--` for others (i.e. world readable).

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#filepermissions

12.25 E0025 – Duplicate names when ignoring letter case for files or directories

As there are operating systems which do not distinguish between `myfile` and `MYFILE` we don't want to have file names in a directory which are the same after converting to lower case.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#filenames

12.26 E0026 – Files not in TDS or different in TDS and non-install tree

The file mentioned in the error message is either not existing in the TDS zip archive, or it is different to the one in the non-install tree.

12.27 E0027 – An I/O error occurred

Some kind of I/O error occurred. If you believe there is an error in `pkgcheck` please contact the author.

12.28 E0028 – A path name in a TDS zip archive must contain the package name

The path names in a TDS zip archive must contain the package name.
Example: Assume a package `somepkg`. Then path names should look like follows:

```
tex/latex/somepkg/somepkg.cls
doc/latex/somepkg/README
source/latex/somepkg/somepkg.dtx
...
```

12.29 E0029 – README file: encoding with BOM detected

A README file should be either ASCII or UTF-8 without BOM(byte order mark)
For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#readme

12.30 E0030 – A symlink was found which points outside of the package directory tree

A symlink must not point to a file or directory outside of the package directory tree.

12.31 E0031 – File name contains invalid UTF-8 character(s)

A file name contains invalid UTF-8 character(s).

12.32 E0032 – Extension expects some file type but got another type

The error shows up when the file has a known extension (e.g., `.pdf`) and the detected type doesn't match. The extension is what drove the expectation.
Example:

```
extension .pdf expects PDF, got Text (LF)
```

12.33 E0034 – Unwanted file detected in the top level directory in TDS zip archive

A top level directory of a TDS archive should only contain certain directories but no files.

12.34 E0035 – Unwanted TDS archive detected in package directory tree

A package directory should not contain a TDS zip archive.

12.35 E0036 – .dtx/.ins files found in wrong directory in TDS zip archive

In a TDS zip archive a `.dtx` resp. `.ins` file must be in a subdirectory of either of `source/` or `doc/` top level directories.

12.36 E0037 – CR line endings detected

The file specified in the error message contains CR line endings. Text files should have UNIX style line endings.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#crlf

12.37 E0038 – File has inconsistent line endings: CR: x, LF: y, CRLF: z

The file specified in the error message contains CR line endings. Text files should have UNIX style line endings.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#crlf

12.38 E0039 – No doc/ directory found in the top level directory of the TDS zip archive

A TDS zip archive is required to contain a top level directory `doc/`.

12.39 E0040 – Too few top level directories in the TDS zip archive

The top level directory of a TDS zip archive must contain at least a `doc` directory and one or more of the following directories: `asymptote`, `bibtex`, `chktex`, `context`, `dvipdfmx`, `dvips`, `fonts`, `hbf2gf`, `makeindex`, `metafont`, `metapost`, `mft`, `omega`, `pbibtex`, `psutils`, `scripts`, `source`, `tex`, `tex4ht`, `texconfig`, `texdoc`, `texdoctk`, `ttf2pk`, `web2c`, `xdvi`, `xindy`,

Any other other directory at the top level is an error.

12.40 E0041 – One or more map file found for the package but none of them is in a path starting with fonts/map/dvips

At least one map file was found which was not in a path starting with `fonts/map/dvips`.

12.41 E0042 – TDS zip archive: duplicate names when ignoring letter case for files or directories

As there are operating systems which do not distinguish between `myfile` and `MYFILE` we don't want to have file names in a directory which are the same after converting to lower case.

For more details refer to:
https://mirror.ctan.org/help/ctan/CTAN-upload-addendum.html#filenames

12.42 E0043 – Symlink found in TDS zip archive

The TDS zip archive contained a symlink which is not allowed.

12.43 E0044 – Wrong permission for directory in TDS zip archive

Directories in TDS zip archive should have `rwx` for the owner and at least `r-x` for others (i.e. world readable).

12.44 E0045 – Wrong permissions for file in TDS zip archive

Files in the TDS zip archive should be world readable.

Binaries should have the x-bit set.

Scripts with a shebang may have the x-bit set but this is not required.

The same for Windows command files which are files ending in one of `.bat`, `.cmd`, `.ps1`, `.nsh` or `.reg`. x-bit may be set but is not required.

12.45 E0046 – File/Directory in TDS zip archive has special bits on

The following occurences will be detected:

- setuid bit (0o4000) on any file
- setgid bit (0o2000) on any file
- sticky bit (0o1000) on any directory

This is an error which usually does not have any impact because unpacker programs usually strip those bits for security reasons. Examples: `unzip` and `7z`.

12.46 E0047 – File/Directory has special bits on

The following occurences will be detected:

- setuid bit (0o4000) on any file
- setgid bit (0o2000) on any file
- sticky bit (0o1000) on any directory

This is an error which should not come up because unpacker programs usually strip those bits for security reasons. Examples: `unzip` and `7z` and also the `unpak` utility used by the CTAN team.