



Linux Ubuntu 22.4.5 Manual Pages on command 'Lintian::Tutorial::TestSuite.3'

\$ man Lintian::Tutorial::TestSuite.3

Lintian::Tutorial::TestSuite(3) Debian Package Checker Lintian::Tutorial::TestSuite(3)

NAME

Lintian::Tutorial::TestSuite -- Quick intro to running the Lintian testsuite

SYNOPSIS

Warning: This document may be out of date.

This guide will quickly introduce you to running the Lintian test suite and some tricks. The Lintian test suite is fairly large and accordingly it can take a substantial amount of time to run. To speed up development, there are various options to limit the tests to run.

If you are looking for a guide on how to write tests, please consult

Lintian::Tutorial::WritingTests.

DESCRIPTION

The Lintian test suite is an extensive collection of various test cases. The test suite is divided into 4 "sub-suites". The majority of tests are currently located in the "tests" sub-suite.

To run the full suite:

```
$ rm -rf debian/test-out; t/bin/build-test-packages; t/bin/runtests
```

While writing a new tag (or check) you probably only want to run a particular (subset of the) test(s). See "Running a subset of the tests" for the available options.

Running a subset of the tests

First, you have to build the test packages with:

```
$ rm -rf debian/test-out; t/bin/build-test-packages;
```

Then, the following options are available:

Running a single test

To run a single test by its name, use:

```
$ t/bin/runtests --onlyrun=test:$name
```

Running all tests for a check

To run all tests for a given check, use:

```
$ t/bin/runtests --onlyrun=check:$name
```

\$check must be the name of a check (it will test for checks/\$check.desc) or "legacy". This will run all tests that start with "\$check-".

Running all tests designed for a specific tag

To run all tests that have a "Test-For" or a "Test-Against" for a given tag, use:

```
$ t/bin/runtests --onlyrun=tag:$name
```

Running tests under coverage

This feature is currently untested.

It is possible to run most of the tests under `Devel::Cover`. This is done by passing `--coverage` to `t/bin/runtests`. Example:

```
$ t/bin/runtests --coverage --dump-logs -j1 -k t debian/test-out
```

Please note that `Devel::Cover` does not seem to handle multiple threads too well. You may see spurious warnings/errors if you run the tests with 2 or more active worker threads.

Caveat 1: Coverage for collections (i.e. programs in collection/) does not seem to work at the moment. Therefore, they often end up with (next to) zero coverage in the generated reports.

Caveat 2: `Devel::Cover` sometimes changes the output of `Lintian` or tools called by `Lintian`. Obviously, this leads to test failures. Therefore, you may see weird test failures (or warnings) when running under coverage.

Collecting the coverage you want in a reasonable time

Collecting coverage is excruciatingly slow. This is not helped by the fact that it becomes unreliable when run under 2 or more threads.

Fortunately, `Devel::Cover` "appends" to its cover database. This allows you to "slowly" build up the coverage database over multiple runs. Example:

```
$ t/bin/runtests --coverage --dump-logs -j1 -k t debian/test-out suite:scripts
```

```
$ t/bin/runtests --coverage --dump-logs -j1 -k t debian/test-out suite:debs
```

```
$ t/bin/runtests --coverage --dump-logs -j1 -k t debian/test-out suite:source
```

...

Or:

```
$ t/bin/runtests --coverage --dump-logs -j1 -k t debian/test-out $check
```

```
$ t/bin/runtests --coverage --dump-logs -j1 -k t debian/test-out legacy
```

SEE ALSO

[Lintian::Tutorial::WritingTests](#)

[Lintian v2.62.0ubuntu2.2](#)

2022-11-09

[Lintian::Tutorial::TestSuite\(3\)](#)