NAME

Test::Lintian −− Check Lintian files for issues

SYNOPSIS

```
# file 1
use Test::Lintian;
use Test::More import => ['done_testing'];
test_load_profiles('some/path');

done_testing;

# file 2
use Test::Lintian;
use Test::More import => ['done_testing'];
load_profile_for_test('vendor/profile', 'some/path', '/usr/share/lintian');
test_check_desc('some/path/checks');
test_load_checks('some/path/checks');
test_tags_implemented('some/path/checks');

done_testing;
```

DESCRIPTION

A testing framework for testing various Lintian files for common errors.

FUNCTIONS

test_check_desc([OPTS,]CHECKS...)

Test check desc files (and the tags in them) for common errors.

OPTS is an optional HASHREF containing key/value pairs, which are described below.

CHECKS is a list of paths in which to check desc files. Any given element in CHECKS can be either a file or a dir. Files are assumed to be check desc file. Directories are searched and all *.desc* files in those dirs are processed.

As the number of tests depends on the number of tags in desc, it is difficult to ''plan ahead'' when using this test. It is therefore recommended to not specify a plan and use **done_testing()**.

This sub uses a Data file (see ''load_profile_for_test ([PROFNAME[, INC...]])'').

OPTS may contain the following key/value pairs:

filter

If defined, it is a filter function that examines `$_` (or its first argument) and returns a truth value if `$_` should be considered or false otherwise. `$_` will be the path to the current file (or dir) in question; it may be relative or absolute.

NB: *all* elements in CHECKS are subject to the filter.

CAVEAT: If the filter rejects a directory, none of the files in it will be considered either. Even if the filter accepts a file, that file will only be processed if it has the proper extension (i.e. with *.desc*).

translation

If defined and a truth value, the desc files are expected to contain translations. Otherwise, they must be regular checks.

test_load_profiles(ROOT, INC...)

Test that all profiles in *ROOT/profiles* are loadable. INC will be the INC path used as include path for the profile.

If INC is omitted, then the include path will consist of (ROOT, '/usr/share/lintian'). Otherwise, INC will be used as is (and should include ROOT).

This sub will do one test per profile loaded.

test_load_checks([OPTS,]DIR[, CHECKNAMES...])
Test that the Perl module implementation of the checks can be loaded and has a run sub.

OPTS is an optional HASHREF containing key/value pairs, which are described below.

DIR is the directory where the checks can be found.

CHECKNAMES is a list of check names. If CHECKNAMES is given, only the checks in this list will be processed. Otherwise, all the checks in DIR will be processed.

For planning purposes, every check processed counts for 2 tests and the call itself does on additional check. So if CHECKNAMES contains 10 elements, then 21 tests will be done (2 * 10 + 1). Filtered out checks will *not* be counted.

All data files created at compile time or in the file scope will be loaded immediately (instead of lazily as done during the regular runs). This is done to spot missing data files or typos in their names. Therefore, this sub will load a profile if one hasn't been loaded already. (see ''load_profile_for_test ([PROFNAME[, INC...]])'')

OPTS may contain the following key/value pairs:

filter
If defined, it is a filter function that examines $_ (or its first argument) and returns a truth value if $_ should be considered or false otherwise. $_ will be the path to the current file (or dir) in question; it may be relative or absolute.

NB: filter is *not* used if CHECKNAMES is given.

CAVEAT: If the filter rejects a directory, none of the files in it will be considered either. Even if the filter accepts a file, that file will only be processed if it has the proper extension (i.e. with *.desc*).

test_tags_implemented ([OPTS,]DIR[, CHECKNAMES...])
Test a given check implements all the tags listed in its desc file. For planning purposes, each check counts as one test and the call itself do one additional check. So if 10 checks are tested, the plan should account for 11 tests.

This is a simple scan of the source code looking asserting that the tag names *appear* (in the actual code part). For a vast majority of Lintian's tags it is reliable enough to be useful. However it has false-positives and false-negatives – the former can be handled via ''exclude-pattern'' (see below).

The DIR argument is the directory in which to find the checks.

CHECKNAMES is a list of the check names. If CHECKNAMES is given, only the checks in this list will be processed. Otherwise, all the checks in DIR will be processed.

The optional parameter OPTS is a hashref. If passed it must be the first argument. The following key/value pairs are defined:

exclude-pattern
The value is assumed to be a regex (or a string describing a regex). Any tag matching this regex will be excluded from this test and is assumed to be implemented (regardless of whether that is true or not).

This is useful for avoiding false-positives with cases like:

```
foreach my $x (@y) {
  tag "some-tag-for-$x", "blah blah $x"
      unless f($x);
}
```

filter

If defined, it is a filter function that examines `$_` (or its first argument) and returns a truth value if `$_` should be considered or false otherwise. `$_` will be the path to the current file (or dir) in question; it may be relative or absolute.

NB: filter is *not* used if CHECKNAMES is given.

CAVEAT: If the filter rejects a directory, none of the files in it will be considered either. Even if the filter accepts a file, that file will only be processed if it has the proper extension (i.e. with *.desc*).

As mentioned, this test assert that the tag name appears in the code. Consider the following example:

```
my $tagname = 'my-tag';
$tagname = 'my-other-tag' if $condition;
```

In this example, this test would conclude that 'my−tag' and 'my−other−tag' are both implemented.

Comment lines are *not* ignored, so comments can be used as an alternative to the exclude-pattern (above).

load_profile_for_test ([PROFNAME[, INC...]])

Load a Lintian::Profile and ensure Data files can be used. This is needed if the test needs to access a data file or if a special profile is needed for the test. It does *not* test the profile for issues.

PROFNAME is the name of the profile to load. It can be omitted, in which case the sub ensures that a profile has been loaded. If no profile has been loaded, 'debian/main' will be loaded.

INC is a list of extra ''include dirs'' (or Lintian ''roots'') to be used for finding the profile. If not specified, it defaults to *$ENV{'LINTIAN_TEST_ROOT'}* and */usr/share/lintian* (in order). INC is ignored if a profile has already been loaded.

CAVEAT: Only one profile can be loaded in a given test. Once a profile has been loaded, it is not possible to replace it with another one. So if this is invoked multiple times, PROFNAME must be omitted or must match the name of the loaded profile.

program_name_to_perl_paths(PROGNAME)

Map the program name (e.g. `$0`) to a list of directories or/and files that should be processed.

This helper sub is mostly useful for splitting up slow tests run over all Perl scripts/modules in Lintian. This allows better use of multiple cores. Example:

```
t/scripts/my-test/
 runner.pl
 checks.t -> runner.pl
 collection.t -> runner.pl
 ...
```

And then in runner.pl:

```
use Test::Lintian;

my @paths = program_name_to_perl_paths($0);
# test all files/dirs listed in @paths
```

For a more concrete example, see t/scripts/01−critic/ and the files/symlinks beneath it.