

NAME

Test::Harness - Run Perl standard test scripts with statistics

VERSION

Version 3.36

SYNOPSIS

```
use Test::Harness;

runtests(@test_files);
```

DESCRIPTION

Although, for historical reasons, the *Test::Harness* distribution takes its name from this module it now exists only to provide *TAP::Harness* with an interface that is somewhat backwards compatible with *Test::Harness* 2.xx. If you're writing new code consider using *TAP::Harness* directly instead.

Emulation is provided for `runtests` and `execute_tests` but the pluggable 'Straps' interface that previous versions of *Test::Harness* supported is not reproduced here. Straps is now available as a stand alone module: *Test::Harness::Straps*.

See *TAP::Parser*, *TAP::Harness* for the main documentation for this distribution.

FUNCTIONS

The following functions are available.

`runtests(@test_files)`

This runs all the given *@test_files* and divines whether they passed or failed based on their output to STDOUT (details above). It prints out each individual test which failed along with a summary report and a how long it all took.

It returns true if everything was ok. Otherwise it will `die()` with one of the messages in the **DIAGNOSTICS** section.

`execute_tests(tests => \@test_files, out => *FH)`

Runs all the given *@test_files* (just like `runtests()`) but doesn't generate the final report. During testing, progress information will be written to the currently selected output filehandle (usually STDOUT), or to the filehandle given by the *out* parameter. The *out* is optional.

Returns a list of two values, *\$total* and *\$failed*, describing the results. *\$total* is a hash ref summary of all the tests run. Its keys and values are this:

<code>bonus</code>	Number of individual todo tests unexpectedly passed
<code>max</code>	Number of individual tests ran
<code>ok</code>	Number of individual tests passed
<code>sub_skipped</code>	Number of individual tests skipped
<code>todo</code>	Number of individual todo tests
<code>files</code>	Number of test files ran
<code>good</code>	Number of test files passed
<code>bad</code>	Number of test files failed
<code>tests</code>	Number of test files originally given
<code>skipped</code>	Number of test files skipped

If `$total->{bad} == 0` and `$total->{max} > 0`, you've got a successful test.

\$failed is a hash ref of all the test scripts that failed. Each key is the name of a test script, each

value is another hash representing how that script failed. Its keys are these:

name	Name of the test which failed
estat	Script's exit value
wstat	Script's wait status
max	Number of individual tests
failed	Number which failed
canon	List of tests which failed (as string).

`$failed` should be empty if everything passed.

EXPORT

`&runtests` is exported by `Test::Harness` by default.

`&execute_tests`, `$verbose`, `$switches` and `$debug` are exported upon request.

ENVIRONMENT VARIABLES THAT TAP::HARNESS::COMPATIBLE SETS

`Test::Harness` sets these before executing the individual tests.

HARNESS_ACTIVE

This is set to a true value. It allows the tests to determine if they are being executed through the harness or by any other means.

HARNESS_VERSION

This is the version of `Test::Harness`.

ENVIRONMENT VARIABLES THAT AFFECT TEST::HARNESS

HARNESS_PERL_SWITCHES

Setting this adds perl command line switches to each test file run.

For example, `HARNESS_PERL_SWITCHES=-T` will turn on taint mode.

`HARNESS_PERL_SWITCHES=-MDevel::Cover` will run `Devel::Cover` for each test.

`-w` is always set. You can turn this off in the test with `BEGIN { $^W = 0 }`.

HARNESS_TIMER

Setting this to true will make the harness display the number of milliseconds each test took. You can also use *prove's* `--timer` switch.

HARNESS_VERBOSE

If true, `Test::Harness` will output the verbose results of running its tests. Setting `$Test::Harness::verbose` will override this, or you can use the `-v` switch in the *prove* utility.

HARNESS_OPTIONS

Provide additional options to the harness. Currently supported options are:

j<n>

Run <n> (default 9) parallel jobs.

c

Try to color output. See *"new" in TAP::Formatter::Base*.

a<file.tgz>

Will use *TAP::Harness::Archive* as the harness class, and save the TAP to `file.tgz`

fPackage-With-Dashes

Set the `formatter_class` of the harness being run. Since the `HARNESS_OPTIONS` is

seperated by :, we use - instead.

Multiple options may be separated by colons:

```
HARNESS_OPTIONS=j9:c make test
```

```
HARNESS_SUBCLASS
```

Specifies a TAP::Harness subclass to be used in place of TAP::Harness.

```
HARNESS_SUMMARY_COLOR_SUCCESS
```

Determines the *Term::ANSIColor* for the summary in case it is successful. This color defaults to 'green'.

```
HARNESS_SUMMARY_COLOR_FAIL
```

Determines the *Term::ANSIColor* for the failure in case it is successful. This color defaults to 'red'.

Taint Mode

Normally when a Perl program is run in taint mode the contents of the PERL5LIB environment variable do not appear in @INC.

Because PERL5LIB is often used during testing to add build directories to @INC `Test::Harness` passes the names of any directories found in PERL5LIB as -I switches. The net effect of this is that PERL5LIB is honoured even in taint mode.

SEE ALSO

TAP::Harness

BUGS

Please report any bugs or feature requests to `bug-test-harness` at rt.cpan.org, or through the web interface at <http://rt.cpan.org/NoAuth/ReportBug.html?Queue=Test-Harness>. I will be notified, and then you'll automatically be notified of progress on your bug as I make changes.

AUTHORS

Andy Armstrong <andy@hexten.net>

Test::Harness 2.64 (maintained by Andy Lester and on which this module is based) has this attribution:

Either Tim Bunce or Andreas Koenig, we don't know. What we know for sure is, that it was inspired by Larry Wall's F<TEST> script that came with perl distributions for ages. Numerous anonymous contributors exist. Andreas Koenig held the torch for many years, and then Michael G Schwern.

LICENCE AND COPYRIGHT

Copyright (c) 2007-2011, Andy Armstrong <andy@hexten.net>. All rights reserved.

This module is free software; you can redistribute it and/or modify it under the same terms as Perl itself. See *perlartistic*.