Why do I need to know how many tests I will be running with Test::More?

Asked 15 years, 9 months ago Modified 13 years, 1 month ago

Viewed 3k times

27

Am I a bad person if I use `use Test::More qw(no_plan)`?

The Test::More POD says

> Before anything else, you need a testing plan. This basically declares how many tests your script is going to run to protect against premature failure...
>
> ```
> use Test::More tests => 23;
> ```
>
> There are rare cases when you will not know beforehand how many tests your script is going to run. In this case, you can declare that you have no plan. (Try to avoid using this as it weakens your test.)
>
> ```
> use Test::More qw(no_plan);
> ```

But premature failure can be easily seen when there are no results printed at the end of a test run. It just doesn't

seem that helpful.

So I have 3 questions:

1. What is the reasoning behind requiring a test plan by default?
2. Has anyone found this a useful and time saving feature in the long run?
3. Do other test suites for other languages support this kind of thing?

perl unit-testing testing

Share

Improve this question

Follow

edited Mar 28, 2009 at 19:04

asked Mar 27, 2009 at 15:59

Eric Johnson

17.9k ● 10 ● 54 ● 60

9 Answers

Sorted by: Highest score (default)

34

What is the reason for requiring a test plan by default?

ysth's answer links to a great discussion of this issue which includes comments by Michael Schwern and Ovid

who are the `Test::More` and `Test::Most` maintainers respectively. Apparently this comes up every once in a while on the perl-qa list and is a bit of a contentious issue. Here are the highlights:

Reasons to not use a test plan

1. Its annoying and takes time.
2. Its not worth the time because test scripts won't die without the test harness noticing except in some rare cases.
3. `Test::More` can count tests as they happen
4. If you use a test plan and need to skip tests, then you have the additional pain of needing a `SKIP{}` block.

Reasons to use a test plan

1. It only takes a few seconds to do. If it takes longer, your test logic is too complex.
2. If there is an exit(0) in the code somewhere, your test will complete successfully without running the remaining test cases. An observant human may notice the screen output doesn't look right, but in an automated test suite it could go unnoticed.
3. A developer might accidentally write test logic so that some tests never run.

4. You can't really have a progress bar without knowing ahead of time how many tests will be run. This is difficult to do through introspection alone.

The alternative

`Test::Simple`, `Test::More`, and `Test::Most` have a `done_testing()` method which should be called at the end of the test script. This is the approach I take currently.

This fixes the problem where code has an `exit(0)` in it. It doesn't fix the problem of logic which unintentionally skips tests though.

In short, its safer to use a plan, but the chances of this actually saving the day are low unless your test suites are complicated (and they should not be complicated).

So using `done_testing()` is a middle ground. Its probably not a huge deal whatever your preference.

Has this feature been useful to anyone in the real world?

A few people mention that this feature has been useful to them in the real word. This includes Larry Wall. Michael Schwern says the feature originates with Larry, more than 20 years ago.

Do other languages have this feature?

None of the xUnit type testing suites has the test plan feature. I haven't come across any examples of this feature being used in any other programming language.

Share Improve this answer

Follow

edited May 23, 2017 at 12:01

Community Bot

1 ● 1

answered Mar 28, 2009 at 7:38

Eric Johnson

17.9k ● 10 ● 54 ● 60

1 I love your summary - very concise yet comprehensive. Could I add it to perl-qa.hexten.net/wiki/index.php/Plan? Unless you want to yourself, of course :-). – Gaurav Mar 30, 2009 at 16:19

2 additionally, every now and again, you don't know how many tests you are going to run. `foreach (@found_thing) { ok( ... ) }`, this isn't (and shouldn't be) common, but I have had it happen. github.com/jberger/Alien-GSL/blob/master/t/locations.t – Joel Berger Nov 7, 2011 at 21:15

In that case, I like to use `Test::More::plan` and declare the plan count when initialization's done and I know what the count is. – Joe McMahon May 12, 2014 at 23:02

10

I'm not sure what you are really asking because the documentation extract seems to answer it. I want to know if all my tests ran. However, I don't find that useful until the test suite stabilizes.

While developing, I use no_plan because I'm constantly adding to the test suite. As things stabilize, I verify the number of tests that should run and update the plan. Some people mention the "test harness" catching that already, but there is no such thing as "the test harness". There's the one that most modules use by default because that's what MakeMaker or Module::Build specify, but the TAP output is independent of any particular TAP consumer.

A couple of people have mentioned situations where the number of tests might vary. I figure out the tests however I need to compute the number then use that in the plan. It also helps to have small test files that target very specific functionality so the number of tests is low.

```
use vars qw( $tests );

BEGIN {
  $tests = ...; # figure it out

  use Test::More tests => $tests;
  }
```

You can also separate the count from the loading:

```
use Test::More;

plan tests => $tests;
```

The latest TAP lets you put the plan at the end too.

Follow

edited Mar 29, 2009 at 12:03

answered Mar 28, 2009 at 3:13

brian d foy
132k ● 31 ● 211 ● 604

Thanks Brian. I like your suggestion for using no_plan until the suite stabilizes. – Eric Johnson Mar 28, 2009 at 19:11

7

In one comment, you seem to think prematurely exiting will count as a failure, since the plan won't be output at the end, but this isn't the case - the plan will be output unless you terminate with POSIX::_exit or a fatal signal or the like. In particular, die() and exit() will result in the plan being output (though the test harness should detect anything other than an exit(0) as a prematurely terminated test).

You may want to look at Test::Most's deferred plan option, soon to be in Test::More (if it's not already).

There's also been discussion of this on the perl-qa list recently. One thread: http://www.nntp.perl.org/group/perl.qa/2009/03/msg12121.html

Share Improve this answer Follow

answered Mar 28, 2009 at 0:18

ysth
98.3k ● 6 ● 125 ● 216

5

Doing any testing is better than doing no testing, but testing is about being deliberate. Stating the number tests expected gives you the ability to see if there is a bug in the test script that is preventing a test from executing (or executing too many times). If you don't run tests under specific conditions you can use the skip function to declare this:

```
SKIP: {
    skip $why, $how_many if $condition;

    ...normal testing code goes here...
}
```

Share Improve this answer

Follow

answered Mar 27, 2009 at 18:20

Chas. Owens

64.9k • 24 • 138 • 231

1 Thanks for your answer. You are pretty convincing... Its just not something I've ever had a problem with -- this check has never saved me. Most of my code uses small classes, so my tests scripts are simple (~15 tests). Maybe I just don't test as much as I should... but thats another question.
– Eric Johnson Mar 27, 2009 at 18:59

3

I think it's ok to bend the rules and use no_plan when the human cost of figuring out the plan is too high, but this cost is a good indication that the test suite has not been well designed.

Another case where it's useful to have the test_plan explicitely defined is when you are doing this kind of tests:

```
$coderef = sub { my $arg = shift; isa_ok $arg, 'MyClas
do(@args, $coderef);
```

and

```
## hijack our interface to test it's called.
local *MyClass::do = $coderef;
```

If you don't specify a plan, it's easy to miss out that your test failed and that some assertions weren't run as you expected.

Share Improve this answer Follow

answered Mar 27, 2009 at 17:03

community wiki
Yann

> I usually only have like 7 tests so its not *that* hard. Just annoying. Your example is interesting. Still it seems like an uncommon scenario. I think the default should be to skip the plan. Then I would think plan is a cool feature. Is that an unreasonable opinion? – Eric Johnson Mar 27, 2009 at 17:50

3

Having explicitly the number of test in the plan is a good idea, unless it is too expensive to retrieve this number. The question has been properly answered already but I wanted to stress two points:

- Better than no_plan is to use `done_testing()`

  ```
  use Test::More;
  ... run your tests ...;
  done_testing( $number_of_tests_run );
  ```

`# or done_testing() if not number of test is known`

- this Matt Trout blog entry is interesting, and rants about adding a plan vs cvs conflicts and other issues that make the plan problematic: [Why numeric test plans are bad, wrong, and don't actually help anyway]

Share Improve this answer

Follow

answered Sep 28, 2010 at 19:02

Pablo Marin-Garcia

4,231 ● 2 ● 35 ● 50

2

I find it annoying, too, and I usually ignore the number at the very beginning until the test suite stabilizes. Then I just keep it up to date manually. I *do* like the idea of knowing how many total tests there are as the seconds tick by, as a kind of a progress indicator.

To make counting easier I put the following before each test:

```
#----- load non-existant record -----
....
#----- add a new record -----
....
#----- load the new record (by name) -----
....
#----- verify the name -----
etc.
```

Then I can quickly scan the file and easily count the tests, just looking for the `#-----` lines. I suppose I could even

write something up in Emacs to do it for me, but it's honestly not that much of a chore.

Share Improve this answer

Follow

answered Mar 28, 2009 at 12:59

Joe Casadonte

16.8k • 11 • 49 • 61

1

It is a pain when doing TDD, because you are writing new tests opportunistically. When I was teaching TDD and the shop used Perl, we decided to use our test suite the no plan way. I guess we could have changed from no_plan to lock down the number of tests. At the time I saw it as more hindrance than help.

Share Improve this answer

Follow

edited Mar 28, 2009 at 3:00

brian d foy

132k • 31 • 211 • 604

answered Mar 27, 2009 at 17:38

Tim Ottinger

1,452 • 9 • 5

1

Eric Johnson's answer is exactly correct. I just wanted to add that `done_testing`, a much better replacement to `no_plan`, was released in Test-Simple 0.87_1 recently. It's an experimental release, but you can download it directly from the previous link.

`done_testing` allows you to declare the number of tests you think you've run at the end of your testing script,

rather than trying to guess it before your script starts. You can read the documentation here.

Share Improve this answer

Follow

answered Mar 30, 2009 at 9:05

Gaurav

1,908 1 19 23