How Do We Assess and Maintain the Health of the Perl 5 Codebase?

Location

```
The Perl Conference: North America
(YAPC::NA::2017)
Alexandria VA
June 19 2017
```

Introduction

In this presentation we discuss the question: *How do we assess and maintain the health of the Perl 5 core distribution?* We'll discuss:

- How do we test Perl?
- How **well** do we test Perl?
- How can we -- meaning: you -- help?

Stages of Testing

There are five stages of testing of the Perl 5 core distribution: testing by ...

1 Individual Contributor

2 Committers

3 Smoke Testing

4 CPANtesters

5 Outer World

We'll consider each in turn.

Individual Contributor

The way anyone can contribute to the Perl 5 core distribution is well described in `pod/perlhack.pod`:

```
$> perldoc perlhack
```

In the interest of time, we won't go into the details of working with the Perl 5 source code. Read perlhack, look at the Bonus Slides or contact me at any time during this conference. For our purpose what is important today is that any

changes to the source code must be accompanied by tests. You can follow Larry Wall's example: Here's the first test committed to Perl back in 1987.

[Screenshot from first Perl commit]

What's even more important is that you run the entire Perl 5 test suite before submitting your patch.

[Screenshot of successful run of 'make test']

Once all tests `PASS`, you add and commit your changes; prepare a patch per the instructions in `perlhack` and attach the patch to an email sent to `perlbug@perl.org`.

The main point is: You run the entire test suite before submitting a patch.

Committers

What happens once you've sent your patch to perlbug?

Your patch will appear in our bug tracker https://rt.perl.org/ and it will also be posted to the Perl 5 Porters mailing list/newsgroup. The Perl 5 Porters are the people who have authorization -- "commit bits" -- to commit code to the Perl 5 repository. It is a committer's job to triage bug reports and requests for changes in the source code. Non-controverial changes -- *e.g.,* spelling corrections -- can be immediately applied to the master branch in the repository. The master branch is known as *blead* -- pronounced "bleed". Anything that needs discussion gets discussed in RT and on the mailing list before it is applied to blead.

The committer may wish to see how well the patch performs on different platforms. In that case the committer can create a *smoke-me* branch in the Perl 5 git repository, apply the patch to the smoke branch, and push the branch to the origin. For example, recently Sullivan Beck sent in a large update to the Locale-Codes library which is distributed with the Perl 5 core. I tested it locally and then pushed a branch out for smoke testing.

```
$> git checkout -b locale-codes

$> git am < locale-codes-3.42-3.51.patch

$> git push -u origin locale-codes:smoke-me/jkeenan/sbeck/locale-code
```

Smoke Testing

So what exactly is smoke testing? It's running the core distribution's test suite on as many different platforms and configurations as possible -- platforms configured by individual members of the Perl community. Smoke testers generally use CPAN library Test-Smoke (http://search.cpan.org/dist/Test-Smoke/) to accomplish

this -- and the fact that Test-Smoke is available on CPAN means you can do this too.

When you install Test-Smoke on your machine, you create three directories:

```
$> ls -l
drwxr-xr-x 27 jkeenan jkeenan 12288 Mar 25 12:57 git-perl
drwxrwxr-x  6 jkeenan jkeenan  4096 Mar 25 13:22 install
drwxrwxrwx 28 jkeenan jkeenan 12288 Mar 25 13:22 perl-current
```

The `install/` directory is where all the configuration and run programs are found. We won't go into the details of configuration here. Suffice it to say that a smoke test run is kicked off by simply saying:

```
$> sh ./install/smokecurrent.sh
```

You can kick off a smoke test run *manually,* as I usually do. I have two virtual machines installed on this machine, each of which holds a different version of the FreeBSD operating system. I try to perform a smoke test run on each every few days.

People with better hardware and sysadmin skills than I often *automate* their smoke tests. You can automate your smoke test runs by putting them in a `cron` job and running them, say, once a day. Or you can figure out a way to listen to the Perl 5 repository and kick off smoke test runs on a per-commit basis.

What Happens Once I Generate a Smoke Test Run?

If you use Test-Smoke in its typical configuration, the results of your smoke test run are sent automatically to http://perl5.test-smoke.org/search. Here is a screenshot of its home page.

[Screenshot of top of the main page.]

The home page reports the results of the most recent smoke test run on each of the platforms we have tested.

[Screenshot of bottom of the main page.]

We can use the search functionality to examine results on particular platforms. Here are some recent results from the FreeBSD operating system.

[Screenshot of a results page for FreeBSD.]

Tony Cook provides us with a more compact tabulation of smoke test results on a per-commit basis at http://perl.develop-help.com.

[Screenshot of blead results.]

At Tony's site we can also search for the results on a particular smoke-me branch. Here are the results on the locale-codes branch mentioned earlier.

[Screenshot of a results page for smoke-me/jkeenan/sbeck/locale-codes branch.]

We Want Perl to Work "Everywhere"

Over the past thirty years Perl has been ported to a wide variety of operating systems. We try to keep Perl alive and kicking on as many OSes as we can for as long as we can -- to End-Of-Life and sometimes beyond. We also try to ensure that Perl builds and tests well on new versions of those operating systems. Having Perl **work** "everywhere" presumes we **test** Perl "everywhere" as well.

We can only accomplish this if we get timely reports as to whether we're passing our tests on these OSes old and new. That means -- believe it or not! -- that we have to test Perl 5 blead on operating systems other than Linux!

Let me give a concrete example of the value of smoke testing which we experienced over the past year.

In early 2016 Karl Williamson did a tremendous amount of work improving Perl's handling of locales. Once we released perl-5.24.0 in May of last year, Karl's work was merged into blead. There were a few test failures reported during smoke testing, but those were quickly corrected.

At that time, however, almost all our smoke test reports were being generated on Linux. For instance, between May and August of last year the number of smoke test reports we received from FreeBSD testers was zero.

Last August, I decided to address this lack of BSD smoke test reports by setting up a virtual machine running FreeBSD on this laptop here. As of last August the latest production release of FreeBSD was version 10.3. I used VirtualBox to set up a FreeBSD-10.3 VM on this machine (which natively runs Ubuntu Linux 16.04).

After getting used to that environment I decided to set up a Test-Smoke configuration inside that VM. After a lot of sweat -- it was August, remember -- I finally got my first FreeBSD-10.3 smoke test report sent off to test-smoke.org. Note that this was the first smoke test report from FreeBSD in at least four months -- and the first report from FreeBSD in the 5.25 annual development cycle.

[Screenshot of first report from FreeBSD-10.3: top only.]

Fortunately, the report was a `PASS`.

In October of last year, however, the FreeBSD team released FreeBSD-11.0 as the latest production release. I realized that I did not know whether Perl 5 blead would build and test correctly on this new version. So I set up a second virtual machine -- this time using VMWare -- and installed FreeBSD-11.0 in that VM. I then set up a Test-Smoke configuration and triggered my first smoke test

run. This was the first smoke test of Perl 5 blead on FreeBSD-11.0. This time, however, the result was `FAIL`.

[Screenshot of first report from FreeBSD-11.0.]

Moreover, the test that failed was `lib/locale.t`. This suggested that the problem might lie in the locale-related commits that Karl Williamson had made *five months before.*

To make a long story short, it took us the better part of two months to figure out what the problems were. If we had been smoke testing Perl 5 blead against FreeBSD-11.0 when it was still in development, we could have addressed the problems much earlier and with much less stress.

What's true of FreeBSD is even more true of other operating systems. Would you be surprised if I said that we are woefully lacking in smoke tests of blead on Windows? On Darwin, OpenBSD or NetBSD? Diversity matters.

CPANtesters

So far the only programs we've discussed have been those found in the Perl 5 test suite itself. But this **primary** test suite, though vast, cannot anticipate every situation in which Perl is actually used. We can, however, use the code which people like you and I have put on CPAN as a **secondary** test suite.

If you go to http://matrix.cpantesters.org/, you see the results of the work of many people who have set up machines that listen for new uploads to CPAN, then test those uploads against various versions of Perl on various operating systems. Generally speaking, a new CPAN upload is tested against all production releases of Perl going back to 5.6 or 5.8. A new CPAN upload is also tested against the latest monthly development "point" release of Perl 5 blead. This means that if a change in the Perl 5 source code has an adverse effect on code anywhere on CPAN, that change should be detectable within a one-month period.

Blead Breaks CPAN (BBC)

If a **recent** change in the Perl 5 source code causes the configuration, build or test of a CPAN distribution to fail, one of the people running CPANtesters rigs -- probably Andreas Koenig or Slaven Rezic -- will file a "BBC" bug ticket. For several years, when I heard the phrase, "BBC ticket," I thought, "Those people at bbc.co.uk in London who use Perl sure find a lot of bugs!" It was several years before I realized that, in this context, "BBC" meant: *Blead Breaks CPAN.*

[Screenshot of typical BBC ticket.]

A BBC bug report is treated very seriously. The Perl 5 Porters have to make a determination as to whether the bug is due to:

- A defect in the code recently committed to Perl 5 blead which causes a failure in valid Perl 5 code in the CPAN module which broke; or
- Code in the CPAN distribution which was substandard and whose defects have been exposed by a valid change in Perl 5 blead; or
- Some combination of the above.

If the defect is, in whole or in part, in Perl 5 blead, then that BBC ticket is listed as a blocker for the next production release of Perl.

We strive for backwards compatibility -- but sometimes we have to break things. Sometimes we have to do security fixes. Sometimes we have to break things to enable Perl's continued growth. Either way, we have to exhaustively test Perl 5 blead against CPAN. In effect, we use CPAN as a proxy for all the Perl 5 code in production -- "in the wild".

The Importance of Testing CPAN against Perl 5 Blead

In the past year, the value of testing Perl 5 blead against CPAN has been proven as never before. As you may know, a security problem was reported which led us to make a major change to perl in the 5.26.0 release. We removed `.` -- the current working directory -- from `@INC` -- the list of directories in which perl searches for modules or programs provided as arguments for the `use`, `require` and `do` built-in functions. There was a tremendous amount of code in the Perl 5 test suite which assumed that `.` was present in `@INC`.

There was an even larger amount of code like that on CPAN. The Perl 5 Porters, along with members of the Perl Toolchain Gang and the authors of CPAN installers like `cpan`, `cpanplus` and `cpanm`, had to consider a number of different short- and medium-term options for removing `.` from `@INC`. Our ability to run the tests of large parts of CPAN on Perl 5 blead was invaluable in shaping our decisions.

Outer World

In recent months we've learned that even testing Perl 5 blead against the entirety of CPAN is not sufficient. Because of Perl's age and ubiquity many applications have been written in it. Some of those applications are widely found in open source software distributions and presume a stable "system perl" or "vendor perl" to run on. In the run-up to perl-5.26.0 we learned of cases where changes in blead would prevent well known utilities from running properly.

One such case concerned `autoconf` a GNU program to automatically detect the configuration settings of a given operating system. `autoconf` depends on Perl -- and other GNU programs in turn depend on `autoconf`. But `autoconf`'s code had one instance of the Perl built-in `do` function which was inconsistent with Perl 5.26's removal of '`.`' from `@INC`. Although the Perl maintainer on Gentoo

Linux had reported that bug to `autoconf`'s maintainers, those maintainers had not made a new release in a long time.

http://lists.gnu.org/archive/html/bug-autoconf/2017-04/msg00002.html

`autoconf` had also not been updated to reflect the fatalization of one warning in perl-5.26.0. In order to prevent failures in `autoconf` from cascading throughout the Linux world, the Perl 5 Porters -- very reluctantly -- had to make one small reversion in the source code relating to the removal of '`.`' from `@INC`.

https://rt.perl.org/Ticket/Display.html?id=130497#txn-1455865

What I am now proposing is a **tertiary** test suite. We should treat applications written in Perl which are widely distributed as packages in the major Linux and BSD distributions as a third-level test suite to be tested against Perl 5 blead. More precisely, when we propose a deprecation or subsequent fatalization of some Perl 5 functionality, we should test it against those applications as soon as possible. We should solicit feedback from the "Perl maintainers" of those distributions as early as possible and take that feedback into consideration when evaluating the change.

Where Can You Help?

You can help us out by taking any of the following steps:

- Subscribe to the Perl 5 Porters mailing list or newsgroup.
- Test Perl 5 blead and proposed branches on your everyday machine.
- Set up a smoke-testing rig with Test::Smoke, particularly on non-Linux platforms.
- Test your CPAN distributions against Perl 5 blead.
- Identify open-source applications written in Perl that we can test against Perl 5 blead.