Name

What's Happening in the core-p7 Branch?

Introduction

The purpose of this presentation is to acquaint you with experimental work being done by Nicolas, myself and others to develop a proof-of-concept of one vision of what Perl 7 might be.

This work was begun some time ago by Nicolas in his own github repository. After Sawyer's presentation at CiC, he moved what he had done into the `core-p7` branch in the main Perl repository. brian.d.foy spotted it there and began submitting pull requests. I spotted those p.r.s and started working in that branch myself.

For reasons that I'll go into later, in the past two days we have resumed working in Nicolas's github repository, so the most up-to-date version of the `core-p7` branch is that found there.

For reasons that I will also go into later, it is likely that we will soon change the name of this branch (in either repository) to something that makes clear that we are not intending this branch to hold the one, true, canonical implementation of Perl 7.

I will also note aspects of the work in the `core-p7` branch that differ either from what Todd has written up in his wiki page, The Proposal for Perl 7 or from what my own current preferences are.

As stated in the branch's README, the branch *"… is experimental and based on top of v5.32.0. It tries to show what a perl binary compiled with strict, warnings and several features like signature, no indirect … (as described in the Proposal for Perl 7) would look like."*

The default functionality in `core-p7` includes:

```
    strict
    warnings
    bitwise
    current_sub
    evalbytes
    fc
    no indirect
    postderef_qq
    say
    state
    switch
    unicode_eval
    unicode_strings (?)
```

Does the `core-p7` branch offer interoperability between Perl 5 and Perl 7. Yes, it does. Let me quote from the README.

"Right now you could change the defaults by using use p5 in a file to avoid enabling v7.0 standards.

…

"The final name could change and we could prefer alternate like use v7 and use v5 or use compat::p5 and use compat::p7. Right now by using p5 and p7 this allows to avoid some technical details and a global replace could be performed later in the development cycle."

Perl 7 Defaults Out of the Box -- No Request Needed

However, I want to emphasize that in this branch right now, you get the Perl 7 defaults right out of the box. You do not have to type anything like `use p7;` or `use v7;` to get those defaults. They're "in the air" there -- just as much as the **absence** of strictures and warnings is "in the air" when you open up a Perl 5 file.

For me -- and here is where I am speaking for myself and not for Todd or Nicolas -- that's perfectly fine. If I'm writing new code I want to write it in Perl 7. I want the Perl 7 defaults to really be there by default. I don't want to have to state in every file that I want Perl 7 defaults. I want to run that against a `perl-7.0` binary and let the compiler tell me if I'm wrong.

As a consequence, in what I have done so far in the `core-p7` branch, I have almost completely ignored the interoperability features. So far I've been mainly concerned with making the test suite strict- and warnings-compliant. To that end in about 80 commits I have yet to type either `use v5` or `use v7`.

That's perfectly fine by me. I don't believe a new major version of a software program should **promise interoperability.** I believe that a new major version of a software program should simply **provide significant interoperability.** I would be perfectly happy with us saying that we will maintain `perl-5.32` for a five- to ten-year period but we will only add new features to `perl-7` and only after we have already released a `perl-7.0.0` which is little more than a clone of `perl-5.32.0`.

The Work on Warnings

As I said, that's my personal opinion. I bring this up mainly to provide background for the **actual coding** I have done in the `core-p7` branch since June 28. By that date Nicolas had made much of the codebase strict-compliant, but he hadn't made it warnings-compliant. So most of what I have done is to modify test files to suppress, capture or avoid warnings.

Suppressing warnings

"Suppressing" warnings means running a test file, discovering where, once warnings are on by default, warnings are emitted where they would not have been when running against Perl 5. For at least twenty years, running tests without turning warnings on was apparently considered standard operating procedure. So many of our older test files were throwing warnings left and right. The least invasive way to do this is to place the code throwing warnings into a block and then call something like `no warnings 'uninitialized';` before the offending statement.

Capturing and testing warnings

There are other situations where the warning is not simply an annoyance. The warning is

something we expect to be generated. And since we expect it to be generated we want to be able to demonstrate that it has been generated. So we capture the warning with something like `$SIG{__WARN__}` and write a test to match the warning we got against the warning we expected.

Avoiding warnings

Finally, there are situations where, when we turn the warning on, we realized that we really were doing the wrong thing all along. We blinded ourselves to that by not running with warnings. In that case, the best thing to do is to re-write the code (as little as possible) to eliminate the condition which was generating the warning.

I've made 77 commits since June 28. My estimate is that two-thirds of them have been to address warnings.

Work Organization in the core-p7 Branch

Now I'd like to say something about how we're organizing our work process in the `core-p7` branch.

My firm belief is that in a software project, the people who are actually doing the work get to determine **how** they are going to do the work. This is particularly true when the people doing the work are volunteers and cannot be threatened with the loss of a paycheck if they organize their work process in a way different from what their paymasters want.

People who are **not** actually doing the work may express their opinions as to the goals of the project. At code review time they can critique the implementation of the project (though nitpicking is not appreciated in experimental work on a proof-of-concept like this. And at delivery time those who have commissioned the project can accept or reject the deliverable. But the people actually doing the work must be the people who determine their work process -- even if that deviates from the norm.

So, even when we were making commits to the `core-p7` branch in the main Perl5 repository, we were doing some things differently from the way we would have been working prior to `perl-5.32.0`. We are logging our issues in the issue tracker in Nicolas's github account. This issue tracker is **not** the place to discuss whether Perl 7 is a good idea or not. It is **not** the place to argue that one should be required to state `use v7` in order to get Perl 7 semantics. (The mailing list is the place for that.) It is, however, the place to either:

- log what we have not yet accomplished in pursuit of the branch's stated goals; or
- report places where what we have done in pursuit of the branch's stated goals has been done incorrectly.

So far we have opened up 81 tickets in our issue tracker. We have already closed 45 of them.

Many of the tickets are organized on a 'per-directory' basis. That is, I have run `make test_harness` against a perl compiled in this branch and chopped that up on the basis of the directory in the core distribution where tests are found. So there's one ticket for all the test output for `dist/Data-Dumper`, another for all the tests under `ext/POSIX`, and so forth. Creating tickets this way should provide a point of entry for people who want to join us in working on this

experimental branch.

Other tickets in our issue tracker are more thematic or topical. They refer to patterns of test failures or warnings that are visible across the test suite as a whole. There are probably a half dozen areas where we still have major problems getting tests passing.

Our research in the `core-p7` branch has also enabled us to identify bugs in blead or other possibilities for immediate improvement there. I've created a label, *discovered-thru-p7-research*, to identify such tickets.

Our research in the `core-p7` branch has also enabled us to identify bugs in blead or other possibilities for immediate improvement there. I've created a label `discovered-thru-p7-research` in the main issue tracker for that purpose.

I would also like to mention that I have submitted a branch for smoke-testing. Though, as expected, many test failures are reported, I am glad to report that `Test::Smoke` handled a smoke-test run with a perl `$VERSION` of `7.0.0` with no hiccups.

We would certainly benefit from having additional people working with us on this experiment. However, I should caution that if you are opposed to Sawyer's vision of Perl 7 -- a vision whose most fundamental premise is the belief that **the Perl programming language is in development mode rather than maintenance mode** -- then you should not work on this branch. You should not even look at this branch. That's because this branch, even though it does not claim to be **the** canonical path toward Perl 7, it does claim to be **a** path to Perl 7 and it explicitly presumes that going forward to Perl 7 is as good thing.

Now let me get back to two items I mentioned earlier. Both of these items concern reinforcing the concept that what we are doing is an experiment.

First, starting on Friday evening we've once again begun doing our work using "/github.com/atoomic/perl/tree/core-p7" in Nicolas's github sitehttps: as the location where we make commits. To see pull requests there and be informed of new issues in that work, you must subscribe to that, which you can do in the same way you would do for any other repository on github.

Conversely, this means that new issues and comments on issues will not be emailed to people following the Issues queue in the main Perl 5 repository. So those 81 bug tickets I mentioned earlier? You won't get updates to them unless you subscribe to them -- but when you do you'll be able to filter them into a different folder in your email client based on their Subject lines.

Along the same lines, the commit-bot in irc.perl.org #p5p does not report commits to Nicolas's repository.

Second, I have been told by Todd that "some people" have been complaining about the name of our branch: `core-p7`. I say that Todd has reported that "some people" are saying this, because they haven't said that to me. Nor have I seen complaints about that on the mailing list or on #p5p (even when the IRC bot was very active there with postings about our commits to the branch when we were still working in the main repository). I don't know whether those "some people"

have read the README for the branch or not.

Whatever. In order to emphasize that our branch is experimental we are considering changing its name to include neither 'core' nor 'p7'. Within a few days we may change the name of that branch to something profoundly self-documenting like `queens_duck`.

In closing, let me just note that I have written another document which offers some ideas as to temporarily changing what we do with our development releases. I hope that in these meetings we get an opportunity to discuss those ideas. However, they are distinct from what's happening in the `core-p7` branch and so are not on the agenda at this moment.

Thank you very much.