Status of Perl 7: A Personal View

This is a slightly revised version of a report I prepared for the Perl Steering Council (PSC) in early February 2021 about the status of the Perl 7 efforts which I have been involved in since the Conference in the Cloud (CiC) in June 2020.

Herein I speak only for myself.

Which "Perl 7"?

When I refer to "Perl 7," I'm referring specifically to the project with the following characteristics:

- A line of development in which the goal would be major version 7 of the `perl` executable, developed according to the goals outlined during "Perl Core" discussions in Summer 2020, as listed here. In particular, strictures and warning would be operative *"starting with line 0."* This is to be distinguished from an alternative approach in which they would become strictures and warnings would be come operative after a major version declaration like `use v7;`. I'll call such an alternative approach *"starting with line 1."*
- The work done in Nicolas Rochelemagne's repository, which I'll refer to as *perl-atoomic*; and
- The project management approach discussed on that repository's wiki and implemented in a roadmap of 16 sequential GitHub Issues.

Our/My Work on Perl 7

The short version of what these Perl 7 efforts were and where they stand …

After two or three false starts, we forked from the Perl 5 core distribution at the v5.33.0 tag. The main branch in perl-atoomic is the `alpha` branch. We created branches for each of the 16 sequential GitHub issues as we came to them and worked on only one objective/branch at a time. We did not merge the branch for any objective into `alpha` until we got all tests passing ("running green").

The first three objectives -- bumping the major version number; strict by default; warnings by default -- were expected to be the most difficult. Objectives 1 and 2 were met and the branches for them were merged into `alpha`. Objective 3 is not yet merged. We are about six test files away from that objective. To see the output of `make test` at the head of the `alpha-03-dev-warnings` branch, get this tarball.

I worked intensely on this project from late June through September and gave four online presentations (Perlmonger groups in Toronto, Houston, Philadelphia and D.C.) about it. My work consisted almost entirely of adapting the test suite to work with the incremented major version number, strict-by-default and warnings-by-default. (The guts changes implementing those objectives were largely atoomic's work.) But the last time I met (virtually) with the other principal developers (Sawyer X, Todd Rinaldo, Nicolas Rochelemagne) was on Oct 07 2020.

Since September, the only other person authoring commits has been John Karr. In December I created several pull requests for work on warnings-by-default. They went unreviewed for over a month. On Jan 24 2021, I went ahead and merged them into the `alpha-dev-03-warnings branch`. The tarball cited above reflects those merges.

Work on this version of Perl 7 can charitably described as "stalled." Although we're only six test files away from Objective 3, that work is non-trivial, as it involves adapting both `B::Deparse` and the debugger to run with warnings-by-default.

Should We Proceed?

So, the question arises: Should we proceed further on this line of development?

My reluctant answer is: **No**.

Lack of support from major Perl contributors

Except for a few situations where we cherry-picked commits from Perl 5 blead, no existing committers or significant authors have come on board with the project since the end of June.

```
    $ git log --reverse b420444857986d91c39ac3187af1d7f6011611e6..HEAD | grep
'^Author:' | sort | uniq -c
       1 Author: Christian Walde <walde.christian@gmail.com>
       1 Author: David Mitchell <davem@iabyn.com>
     636 Author: James E Keenan <jkeenan@cpan.org>
      27 Author: John Karr <brainbuz@brainbuz.org>
       2 Author: Karl Williamson <khw@cpan.org>
       7 Author: Nicolas R <atoomic@cpan.org>
      83 Author: Nicolas R <nicolas@atoomic.org>
       9 Author: Todd Rinaldo <toddr@cpan.org>
       1 Author: Veesh Goldman <rabbiveesh@gmail.com>
      13 Author: oodler <oodler@cpan.org>
       2 Author:    ℕicolas ℝ <nicolas@atoomic.org>
    $ git log --reverse b420444857986d91c39ac3187af1d7f6011611e6..HEAD | grep
'^Author:' | wc -l
         782
```

None of the people who were not committed to this vision of Perl 7 by early July and who participated in the "Perl Steering Committee" Zoom meetings in July and August have come on board. Since October, virtually no one has asked me, "Say, Jim, how's Perl 7 doing?" To put it simply, notwithstanding an estimated 1000 person-hours of work, this vision of Perl's future has not caught fire.

On the other hand, we should note that none of the most prominent critics of this vision of Perl 7 have put in any significant time or effort at implementing a *different* version of Perl's future. In particular, AFAIK none of the people who proposed "defaults from line 1" have written any code to demonstrate the feasibility or advantages of that approach. So our partially implemented vision of Perl 7 is the only one on the table so far.

I should note that one aspect of our Perl 7 work has become reflected in ongoing Perl 5 development. It is now generally accepted that all dual-life code in the core distribution (`dist/` and `cpan/`) should be strict- and warnings-compliant. Todd and Nico have had good results

persuading authors of CPAN-upstream authors to accept patches to achieve this compliance; that work has then been synched into blead. The work we've done in Perl 7 (modulo one failing test for `Storable`) shows that **all** of the `.pm` and `.pl` files that ship with core can achieve both strict- and warnings-compliance.

Perl as Unix utility versus Perl as Application Development Language

I'm going to set aside the large number of *"We weren't consulted before this was announced"* responses to Sawyer's CiC proposal. I can understand the feelings behind these responses, but here I want to focus on the technical.

The strongest criticism of the approach to Perl 7 embodied in Sawyer's CiC presentation and in our subsequent work in perl-atoomic was that changing Perl's default behavior would completely disrupt the command-line use of the `perl` executable to run "darkpan" programs written since 1994.

On the one hand, many people who consciously think of themselves as being "in the Perl community" and a (dwindling) number of organizations (for-profit, not-for-profit, government) build complex web applications in Perl.

On the other hand, an unknown but non-trivial number of people use the perl executable in pipelines essentially as a more powerful shell. A program written in Perl which, say, is part of Debian's `apt` suite is expected to behave in exactly the same way for decades. That implies highly consistent default behavior. The same thing is true in OpenBSD, where `perl` is core in much the same way as `ls` or `cat` is. In OpenBSD `perl` is not maintained as a package in the `lang` category as is, say, Python.

Perl 5's greatest strength since 1994 has been that it is both a language in which to write complex applications and a language to use on the command-line. The greatest limitation to Perl 5's long-term viability since 1994 has been that it is both a language in which to write complex applications and a language to use on the command-line. Perl's greatest strength and greatest limitation are exactly the same.

An anecdote to illustrate the foregoing

<anecdote>

I came face to face with this second use of perl in a surprising way during the past seven months. I was preparing to teach an online course in Modern Money Theory for the Henry George School of Social Science in New York City in November and December. I co-taught a similar course in fall 2019, during which I had to accommodate my teaching partner's preference for using Google Slides. In 2020 I was teaching solo, so I used the same program I've used since 2004: MJD's `text2slide` program -- a program which he himself adapted from one written by Sean Burke in the 1990s. Unglamorous, but it got the job done.

`text2slide` takes a single plain-text file, images in a subdirectory, and generates a series of linked HTML pages from those inputs. When I run the slides, I'm clicking from one HTML page

to another; I'm not clicking through pages in a PDF. I've never had any complaints about that at Perl conferences, and no one to whom I've ever sent a tarball of the slides has ever been unable to click through the HTML either.

However, for this course I anticipated that my students (or, at least, the small fraction who would be motivated enough to want to go through the slides on their own) would be more comfortable downloading a single, multi-page PDF and clicking it through that. To make a long story short: I found the solution in another MJD program so obscure that it's not on CPAN, nor is it on MJD's plover.com. I downloaded it, peeked at the code, shuttered at the late-1990s non-strictness -- but it worked. The PDF slides were not attractive, but they sufficed.

</anecdote>

To summarize: I don't think that the benefits of bumping `perl` to a new major version in which strictures and warnings are on by default outweigh the likely costs.

Is there a path forward?

Yes, possibly.

`perl` should have children

The next generation for Perl should be child languages rather than a new major version of the `perl` executable which changes its default behavior in a way many in the information technology world will find objectionable.

Such child languages will have names different from their parent's. These child languages will focus on Perl's use in building complex applications and be willing to sacrifice some of `perl`'s role in Unix pipelines. Nor will such child languages be required or expected to support 35,000+ CPAN libraries "out of the box."

The line of approach suggested by Sawyer at the 2020 CiC could, of course, be one of these children. So could a line of approach which does not attempt to achieve strict- and warnings-by-default but which does seek to implement the other dozen objectives which we haven't gotten to in Perl 7. So could a line of approach whose focus is "great subroutine signatures or else!"

Which children will survive?

Whether these children survive pregnancy, birth and infancy will depend on whether they have organizational support behind them. Whether or not an organization is willing to write a new application (or extend an existing one) in a particular Perl child *and put that application into production* is the acid test. If an organization is willing to put such an application into production, it has to pay people to maintain that application. A language in which people are hired to work has a chance at adolescence and adulthood. A language in which no one is being paid to work will die in infancy.

Suppose, for example, that Booking, cPanel and Fastmail all wanted to continue to develop web applications in Perl (broadly speaking) but all wanted really good subroutine signatures --

something that necessarily comes at the expense of pre-5.22 prototypes. If they collaborated, could they could produce such a language? Yes, undoubtedly, and probably in less than twelve months.

Those children of Perl that survive will be those that meet real organizational production needs and which ignore all the whining on https://www.reddit.com/r/perl/ and #p5p.

James E Keenan
March 25 2021