Non-Sequential

EPIGRAPH

"This is Perl. When in doubt, use a hash."

--attributed to Leon Brocard

DESCRIPTION

This document provides a different approach to preparing Perl development releases during what I will call the **Interregnum**. The Interregnum is the period commencing with the production release of `perl-5.32.0` and terminating with the production release of `perl-7.2.0`. It may -- or may *not* -- include releases of `perl-5.34.0` and `perl-7.0.0`. It does **not** include maintenance releases of `perl-5.32`, which procede as previously, with priority to security releases.

What Is a Non-Sequential Development Release of Perl?

To answer this question we first have to describe the way in which Perl 5's release process, up until now, can be characterized as **sequential**.

Our current, sequential development pattern

The current cycle works like this:

- The pumpking issues a production release of `perl-5.NN.0`, where `NN` is even. The release has a dual form:
  1. A tarball is uploaded to CPAN for worldwide distribution.
  2. The commit in `blead` where the tarball has been generated is tagged `v5.NN.0`.
- The pumpking makes a few metadata-ish commits to `blead`. Ideally, at this time `blead` is frozen. Only the pumpking makes commits to `blead` and none of those commits may have any impact, intended or unintended, on the functionality of `perl-5.NN.0` or its test suite.
- The pumpking changes `5.NN` to `5.NN+1` at important points in the codebase and issues the first development release in the new development cycle, `perl-5.NN+1.0`, where `NN+1` is odd. This release also takes a dual form: an upload to CPAN and a tag in `blead`.
- `blead` is now unfrozen. All commit-bit holders are free to commit code to `blead` that either they themselves have written or that others have submitted via patches or pull requests. Some commits may be minor -- equivalent to a mechanic tightening the screws on the undercarriage of a truck. Others may be major -- equivalent to introducing a new transmission design for that truck.

  From the point of `blead`, all commits are of equal importance. All committers are

presumed to exercise due diligence and courtesy.

The pumpking is not expected to rule on whether a particular commit should go in to `blead` or not. At least one former pumpking has said, *"I go where the patches take me."* So once blead is unfrozen, there is no specific plan for what will go into the next production release (or even the next development release) -- with the possible exception of scheduled deprecations and fatalizations. Hence, this development is largely `unplanned`.

- On or about the 20th day of the month following `perl-5.NN+1.0` a commit-bit holder acting in the role of **release manager** creates the next monthly development release. Once again, this release has a dual form: a tarball named `perl-5.NN+1.1` and a tag in the repository `v5.NN+1.1`. For at least the first six months of an annual development cycle, the release manager exercises no discretion as to what goes into that release. He or she simply tars up blead, tags it and ships it.

  The cycle repeats on a monthly basis.

- This development is **cumulative**. Unless particular commits are reverted, all the functionality that was released in `perl-5.31.1` is still there a month later in `perl-5.31.2`, though new functionality may have been added.

- The cumulative nature of this development is reflected in the fact that release numbers and tags are **sequential** in character. Release `perl-5.31.2` is understood to unambiguously mean "the first monthly release of `perl-5.31.1`." Same thing for tag `v5.31.2` with respect to `v5.31.1`. So we can think of the natural number in the third, "patch version" position in the release name or tag as being an index into an array. Array indices start at 0 and proceed upwards, though in this case we hope that the patch version never goes higher than 11.

- About eight months into the development cycle committers are supposed to begin heeding various freeze points and avoid making certain kinds of commits to `blead`.

- About ten months into the cycle the pumpking must reappear on the scene to start preparing **release candidate** (`RC`) releases in preparation for the next production release. Generally speaking, whatever functionality and code has been accumulating over the past year appears in `perl-5.NN+2.0-RC0`. The pumpking is required to play the role of "senior release manager," but he or she is not required to play the role of "editor."

- At a certain point, the pumpking issues the next production release.

What would it mean for a development pattern to be non-sequential?

In a non-sequential development pattern, the third, "patch version" position in a release name or tag would be treated as a hash key rather than as an array index. This would be so notwithstanding the fact that the characters in the third position would still be positive numbers rather than arbitrary strings. Each such key would signify a continuous stream of development

beginning with `5.NN+1.0`, taking place in a **non-blead** branch and **ending** with the implementation of a specific set of features in a release entitled `perl-5.NN+1.1` and tag <v5.NN+1.1> -- but the final `.1` there would not be treated as "first in an ordered list" but simply as shorthand for "first dev release to have implemented a specific set of new functionality."

In a **different**, non-blead branch, someone **else** would start with `5.NN+1.0` and work on a **different**, specific set of features. That would culminate in `5.NN+1.2` and tag `v5.NN+1.2`. But once again, the final `.2` would not imply "includes the features that were in `.1` plus another month's work." It would imply "a different set of features, implemented from the same starting point (`5.NN+1.0`) but ending in a different place. And I do mean **ending**. Once a dev release was cut from that branch, the branch would be closed to further commits.

Each of these would be development releases of Perl, but they would not be thought of as cumulative, sequential or unplanned. The lead developers for each track would have to announce in advance what they were trying to accomplish in that branch. As they near the end of development of that functionality, they would have to demonstrate that they had achieved their goal and that all the functionality that was in `perl-5.NN+1.0` (or, in effect, `perl-5.NN.0`) was still intact.

Development would therefore be **non-cumulative, non-sequential, but planned**.

A tarball would be released, but not necessarily on a monthly basis. It would be released when the branch lead developer had persuaded the pumpking that the code in the branch deserved exposure to the wider world. The pumpking would not be required to promise that the new functionality would appear in the next production release of Perl.

The Perl community's collective QA infrastructure would have to be modified to accept non-sequential development releases. But these modifications -- to test-smoke.org (currently not operational), CPANtesters, my own CPAN-River-3000 testing would be a SMOP.

At a certain point, the pumpking would have to step forward and make decisions as to which of these various development branches would be cherry-picked for the next production release.

Why Should We Consider Non-Sequential Development Right Now?

We are in the **Interregnum**. Even among those community members who are not opposed to Sawyer X's vision for Perl 7, there is wide disagreement as to what functionality should be included in development releases (5.33 or 7-something?), production releases (Will there be a 5.34? 5.36? What will 7.0 signify as distinct from both 5.32 and 7.2?)

I argue that trying to stick rigidly to our established development pattern is neither wise nor necessary right now. We need to have a number of different strands of development proceeding simultaneously. Then we will need to come back together to sew those strands back together -- a proces that must take place under the pumpking's direction.

Concretely:

- Sawyer X should revert any commits to blead that have been made since the release of `perl-5.32.0` which could have any impact whatsoever on `perl`'s functionality and issue

a `perl-5.33.0` tarball and `v5.33.0` tag. That release will serve as the **baseline** for all future development work.

- `blead` will be frozen until such time as we are ready to resume our current development process -- what I have characterized as a **cumulative, sequential and unplanned** process. Only the pumpking will be permitted to make commits to `blead`.
- People who want to introduce new or modified functionality will announce their plans to do so. If they have commit bits, they can create a branch in the Perl repository on github and conduct development there. They will solicit pull requests into that branch -- not into `blead`. If they do not have commits, they will be encouraged to fork the main Perl repository into their own github accounts, then create new branches starting at the `v5.33.0` tag. Either way, they are free to organize and supervise the development process in their own tracks as they see fit. That may mean, for example, having issues filed in a location other than https://github.com/Perl/perl5/issues.
- In this approach, for example, Paul "LeoNerd" Evans could start at `v5.33.0` and implement `try/catch` functionality. Karl Williamson would also start at `v5.33.0` and implement the next stage of his regex-improvement project. Nicolas R and I would rebase our `core-p7` branch on `v5.33.0` and continue our experimental exploration of "many-more-features-on-by-default." And one other branch would start at `v5.33.0` and be the location for commits of the mechanic-tightening-the-screws-on-the-truck nature I described earlier.
- As each track comes close to achieving its stated goals, the branch manager would ask for review from the pumpking. (Since all this development will be taking place out in the open on github, the community and the world will have had ample opportunity to kibbitz.) If the pumpking gives a thumbs-up, a tarball will be created named with the next available positive number. That tarball will be uploaded to CPAN and will therefore be available for CPANtesters, CPAN-River-3000 and exploration on individual and enterprise codebases. The branch will be tagged at that point but will **not** be merged to blead.

  Whatever is the next feature that achieves that maturity and pumpking approval will get a tarball and tag with the next available positive number. I suspect that over a 12-month period we might get 5 or 6 of these development releases.
- When we have achieved more consensus on questions like:
  - Will there be a 5.34 release?
  - Will Perl 5 be operable from within Perl 7? If so, how?
  - Which specific features will be turned on by default in Perl 7?
  - Do we have a Perl 7 stable enough to release?

  … then we can make decisions on how these tracks will be merged back into a "master" branch. Each of these tracks will have been submitted to the world for experimental use and review. So we won't be flying in the dark. We will have a good idea of what we are promising to maintain for a 5- to 10-year period in major version 5. And we will have a good idea of what major version 7 will look like in its first year of existence.

This is Perl. When in doubt, use a hash.