# Perl 7: An Opinionated Introduction

James E Keenan
CPANID: jkeenan
irc.perl.org: kid51

DC Perlmongers
October 6 2020

Press N for the next page, B for back.

## The Three Big Questions

- When it comes to change ...
    - Should we?
    - Can we?
    - If we can, how should we?

## A Brief History of Perl Development

- 1987: Perl
- 1994: Perl 5
- 2000: Larry Wall announces quest for Perl 6
- 2014: Perl 6 (somewhat)
- 2018: Perl 6 renamed Raku

## What Happened to Perl 5 While Perl 6 Was in Gestation?

- July 2002: 5.8.0
- December 2007: 5.10.0
- April 2010: 5.12.0
- Era of the pumpkings
  - Jesse Vincent (5.12, 5.14) (2010-2012)
  - Ricardo Signes (5.16, 5.18, 5.20, 5.22, 5.24) (2012-2016)
  - Sawyer X (5.26, 5.28, 5.30, 5.32) (2016-present)

## Role of Pumpking

- Chief Release Manager
- Chief Arbiter of Disputes
- NOT Chief Coder
- NOT Chief Language Designer
    - "I go where the patches take me"

## Perl 5 Development Cycle

- Spring: annual production release
    - 5.32.0
- Opening of next development cycle
    - 5.33.0
- Monthly development releases
    - Code freezes phased-in in late winter
- Maintenance production releases

## Perl 5 Development Process

- Cumulative
- Sequential
- Unplanned

An Alternative Approach to Perl Core Development

## "Perl Is Dead"

- A meme you may have noticed
  - Orlando Perl Workshop, January 2013
- 2012: Lots of job postings on jobs.perl.org
- 2016: Few job postings on jobs.perl.org
- Few new projects being written in Perl 5
  - So little reason to hire people as "Perl programmers"
  - Little reason for younger people to study Perl 5

## We’ve Got to Try Something Different

- With Perl 6 name change, "7" becomes available as next major version
- Sawyer’s plan: Seize this opportunity
  - Perl must be in development mode, not just in maintenance mode
  - Establish notion that Perl will continue to evolve
  - A major production release permits changes in default behavior

## Should We Change?

- Many important people in the Perl community say: No!

continued...

## Should We Change?

- Many important people in the Perl community say: No!
- `WARNING! What follows may be exaggerated for rhetorical effect!`

continued...

## Should We Change?

- Many important people in the Perl community say: No!
- `WARNING! What follows may be exaggerated for rhetorical effect!`
- Perl is not like other programming languages
    - It’s also a (Unix) command-line utility, like sed or awk

continued...

## Should We Change?

- Many important people in the Perl community say: No!
- `WARNING! What follows may be exaggerated for rhetorical effect!`
- Perl is not like other programming languages
    - It’s also a (Unix) command-line utility, like sed or awk
- Perl has always promised extensive backwards compatibility
    - A change in default behavior would reneg on this promise

continued...

## Should We Change?

- Many important people in the Perl community say: No!
- `WARNING! What follows may be exaggerated for rhetorical effect!`
- Perl is not like other programming languages
  - It’s also a (Unix) command-line utility, like sed or awk
- Perl has always promised extensive backwards compatibility
  - A change in default behavior would reneg on this promise
- Stability above all else!

continued...

## Should We Change?

- Many important people in the Perl community say: No!
- `WARNING! What follows may be exaggerated for rhetorical effect!`
- Perl is not like other programming languages
    - It's also a (Unix) command-line utility, like sed or awk
- Perl has always promised extensive backwards compatibility
    - A change in default behavior would reneg on this promise
- Stability above all else!
- Prioritize existing code and existing users over new code and new users

continued...

## Should We Change?

- Many important people in the Perl community say: No!
- `WARNING! What follows may be exaggerated for rhetorical effect!`
- Perl is not like other programming languages
  - It’s also a (Unix) command-line utility, like sed or awk
- Perl has always promised extensive backwards compatibility
  - A change in default behavior would reneg on this promise
- Stability above all else!
- Prioritize existing code and existing users over new code and new users
- Otherwise ... "Major Reputational Damage!"

## There Has Been No Decision Yet That We Should Change

- Governance discussions have superseded discussions about Perl’s future direction

continued...

## There Has Been No Decision Yet That We Should Change

- Governance discussions have superseded discussions about Perl’s future direction
- But, in the meantime, some of us are exploring whether we can change

## What Do We Mean by Default Behavior? (1)

[Live demo]

- `p5-01.pl` - A typical Perl 5 program. What is its behavior?
- `p5-02.pl` - Another typical Perl 5 program. What happens when we run it?
- `p7-01.pl` - A typical Perl 7 program. What is its behavior?
- `p7-02.pl` - Another typical Perl 7 program. What happens when we run it?

continued...

## What Do We Mean by Default Behavior? (1)

[Live demo]

- `p5-01.pl` - A typical Perl 5 program. What is its behavior?
- `p5-02.pl` - Another typical Perl 5 program. What happens when we run it?
- `p7-01.pl` - A typical Perl 7 program. What is its behavior?
- `p7-02.pl` - Another typical Perl 7 program. What happens when we run it?
- `p5-03.pl` - Like the second Perl 5 program, but modified to `use strict;`. What happens when we run it?
- `p5-04.pl` - Like the third Perl 5 program, but with corrections. What happens when we run it?
- `p7-03.pl` - Like the second Perl program, but with corrections. What happens when we run it?

## What Do We Mean by Default Behavior? (2)

- The way the progam is expected to behave before you’ve written a single line of code
  - Behavior at "line 0"
- Perl 5 has 1994 defaults
- We have "best practices" -- but they’re not default behavior
  - You have to specially request them

```
use strict;
```

## What Do We Mean by Default Behavior? (3)

- We’ve added many nice features over the years, but they’re not default, either
- You have to specially request them

```
use feature ’say’;
```

- Or you have to request a specific (minor) version of Perl that has a particular feature "bundled" with it

```
use 5.14.0;
```

## Problems with 1994 Defaults

- New programmers forced to add boilerplate code to start of all programs

continued...

## Problems with 1994 Defaults

- New programmers forced to add boilerplate code to start of all programs
- Perl programmers don’t learn "recently" added features
  - I’ve never used any functionality that wasn’t there in May 2011 (5.14)

continued...

## Problems with 1994 Defaults

- New programmers forced to add boilerplate code to start of all programs
- Perl programmers don’t learn "recently" added features
    - I’ve never used any functionality that wasn’t there in May 2011 (5.14)
- CPAN authors: If you want to support all Perl versions since 5.8,
    - ... you can’t use any functionality introduced since 2002

continued...

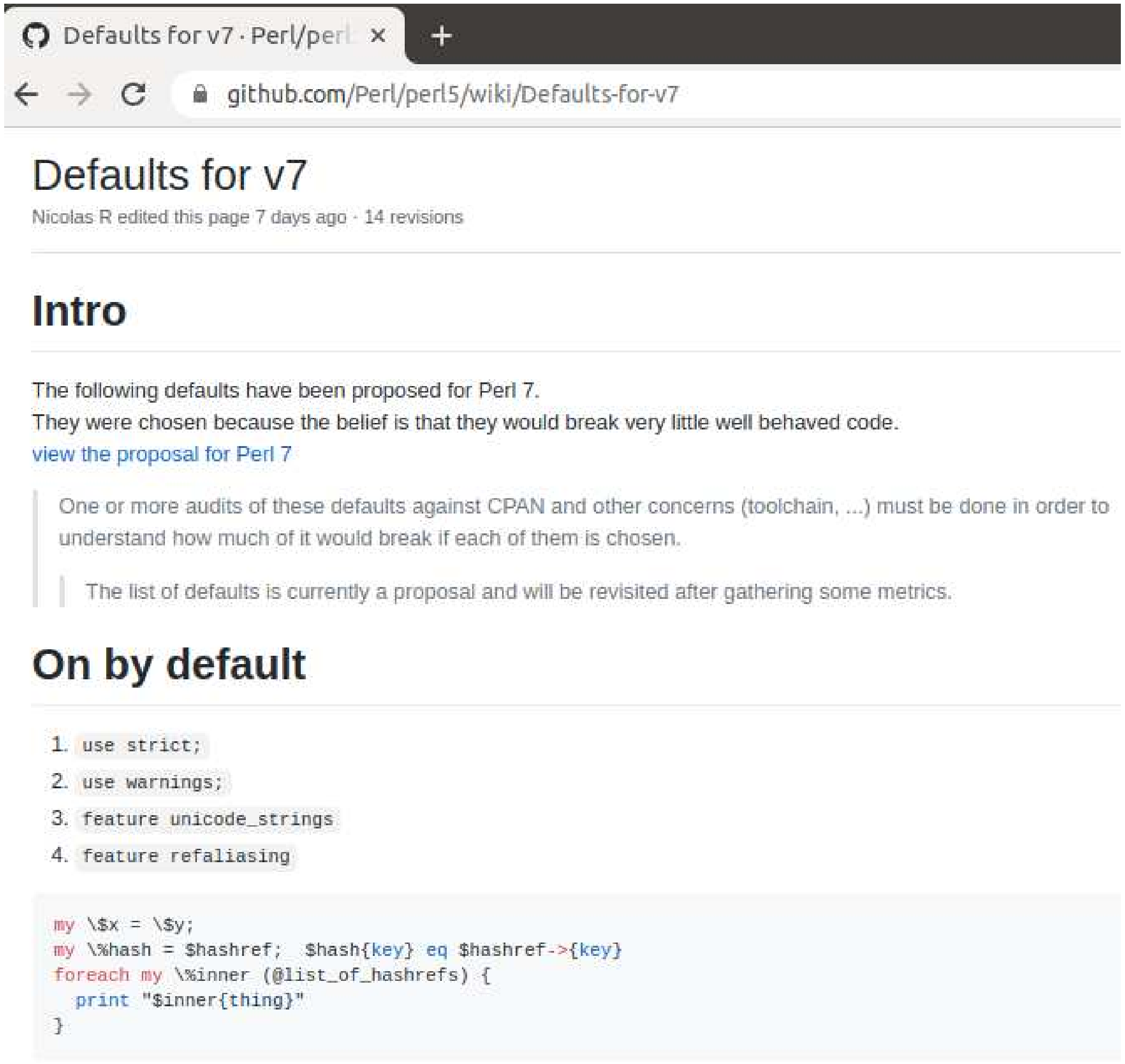
## Problems with 1994 Defaults

- New programmers forced to add boilerplate code to start of all programs
- Perl programmers don’t learn "recently" added features
  - I’ve never used any functionality that wasn’t there in May 2011 (5.14)
- CPAN authors: If you want to support all Perl versions since 5.8,
  - ... you can’t use any functionality introduced since 2002
- Language itself does not enforce good programming practices
  - Feature or bug?

## Perl 7: 2020 Defaults (or, One Proposal For 2020 Defaults)

- `use strict;` by default
- `use warnings;` by default
- The exact list of what will be default behavior in Perl 7 is evolving

Wiki: Perl 7 Defaults


Defaults for v7 · Perl/perl
github.com/Perl/perl5/wiki/Defaults-for-v7

## Defaults for v7

Nicolas R edited this page 7 days ago · 14 revisions

### Intro

The following defaults have been proposed for Perl 7.
They were chosen because the belief is that they would break very little well behaved code.
view the proposal for Perl 7

> One or more audits of these defaults against CPAN and other concerns (toolchain, ...) must be done in order to understand how much of it would break if each of them is chosen.
>
> > The list of defaults is currently a proposal and will be revisited after gathering some metrics.

### On by default

1. `use strict;`
2. `use warnings;`
3. `feature unicode_strings`
4. `feature refaliasing`

```
my \$x = \$y;
my \%hash = $hashref;  $hash{key} eq $hashref->{key}
foreach my \%inner (@list_of_hashrefs) {
  print "$inner{thing}"
}
```

## Perl 8: 202? Defaults

- The most important feature of Perl 7 is that there will be a Perl 8

continued...

## Perl 8: 202? Defaults

- The most important feature of Perl 7 is that there will be a Perl 8
- The language will evolve further
- Defaults will get better

continued...

## Perl 8: 202? Defaults

- The most important feature of Perl 7 is that there will be a Perl 8
- The language will evolve further
- Defaults will get better
  - Indirect object syntax only upon request
    - `no feature 'indirect';` becomes default
    - Removes a blocker to adding new functionality to core

continued...

## Perl 8: 202? Defaults

- The most important feature of Perl 7 is that there will be a Perl 8
- The language will evolve further
- Defaults will get better
    - Indirect object syntax only upon request
        - `no feature 'indirect';` becomes default
        - Removes a blocker to adding new functionality to core
    - Much better subroutine signatures
        - Likely to revolutionize the "look and feel" of writing Perl
        - ... though at the expense of old prototype behavior

## Perl 8: Should We?

- Again, some say: No!
- Major version increments pose burden on certain users, e.g., OS packagers
    - Especially if more than once a decade

## The Argument for Moving Forward

- Was made by Damian Conway on perl.com this past week

Damian Conway on Perl 7

## What do you think about “Perl 7” currently being discussed widely?

I think it’s great to see Perl moving out from under the lingering ghost of “Perl 6”. And to see such a strong statement of positive forward motion, hopefully without too much of the attendant disruption of breaking vast swathes of existing code.

And, more importantly, I think it’s vital that the widely accepted boilerplate components of a modern Perl program (`use strict`, `use warnings`, declarative subroutine parameters, postfix dereferencing) should be turned on by default under new versions of Perl. And, of course, that some of the especially problematic vestigial components (indirect-object method calls, implicit hash-key concatenation, bareword filehandles) should be deprecated with extreme prejudice.

Of course, a plan this bold and this unanticipated will inevitably create anxiety and raise dissent. And not all those fears and disagreements will be misplaced. Even so, the very best thing about Perl 7 (whatever it ultimately proves to be) is that even just the idea of making a major version bump to usher in such fundamental changes is generating a huge amount of productive discussion and debate with the Perl community, and injecting a vast quantity of new energy, which must surely eventually lead to a better outcome, a better path forward for Perl.

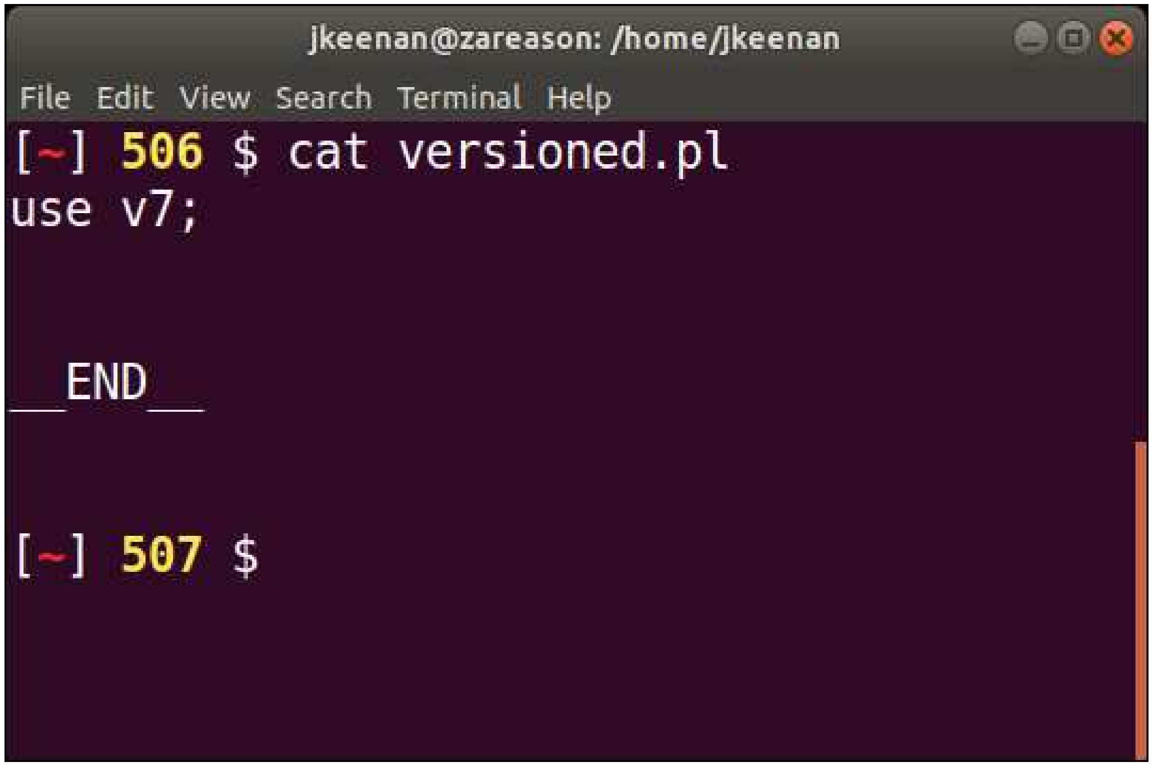
## What Projects Should I Write in Perl 7?

- Assuming that the decision is to change rather than not change ...

continued...

## Perl 7: 2020 Defaults: A Different Proposal

- Others argue that we should not change the 1994 defaults, but ...
  - We should establish that using version declarations is best practice
  - Target "line 1" rather than "line 0"


jkeenan@zareason: /home/jkeenan
File Edit View Search Terminal Help

```
[~] 506 $ cat versioned.pl
use v7;


__END__


[~] 507 $
```

## What Projects Should I Write in Perl 7?

- Assuming that the decision is to change rather than not change ...

continued...

## Do I Need to Re-write Existing Perl 5 Applications in Perl 7?

- No

continued...

## What Projects Should I Write in Perl 7?

- Assuming that the decision is to change rather than not change ...
- New projects

## Do I Need to Re-write Existing Perl 5 Applications in Perl 7?

continued...

## Do I Need to Re-write Existing Perl 5 Applications in Perl 7?

- No

continued...

## Do I Need to Re-write Existing Perl 5 Applications in Perl 7?

- No
- Continue to run them against a `perl-5.xx` executable
  - Consider upgrading them to `perl-5.32`
  - ... which will get maintenance releases for at least 5 years

## How Can I Prepare for Perl 7?

- Do you have a test suite?

continued...

## How Can I Prepare for Perl 7?

- Do you have a test suite?
- Follow best practices

continued...

## How Can I Prepare for Perl 7?

- Do you have a test suite?
- Follow best practices
- If you’ve been following best practices for the past 15 years, 99% of your upgrade problems are already solved
    - `use strict;`
    - `use warnings;`
    - Avoid indirect object syntax
    - Don’t depend on hash keys being returned in a predictable order (security bug fixed in 5.18 (2013))
    - Don’t depend on presence of `.` in `@INC` (security bug fixed in 5.28 (2018)).

## How Can I Prepare for Perl 7?

- Do you have a test suite?

continued...

## How Can I Prepare for Perl 7?

- Do you have a test suite?
- Follow best practices

continued...

## Junk Your Old Prototypes (1)

- Have Perl 5 prototypes ever really been all that useful?
  - Can tell you if you’ve provided the wrong number of mandatory arguments, but ...
  - Very limited type-checking
    - All scalars look alike, whether strings, numbers or references
    - Won’t warn if mandatory argument is undefined
  - Do not preclude need to unpack `@INC`
  - Do not declare named variables for use within subroutine, as signatures would

## Junk Your Old Prototypes (2)

- Were they introduced merely to shut up people coming from other languages who expected them?

continued...

## Junk Your Old Prototypes (2)

- Were they introduced merely to shut up people coming from other languages who expected them?
- In past three months I've learned about their most important use case:

```
use Test::More;

ok("cogito ergo sum",
   "Unit test written with parentheses");

ok "caveat emptor",
   "Take advantage of fact that Test::More::ok() is prototyped -- Look Ma, no parens!"
```

- Prototypes let you define functions that look like Perl built-ins
    - ... and therefore don't require parentheses around their arguments

## Junk Your Old Prototypes (2)

- Were they introduced merely to shut up people coming from other languages who expected them?

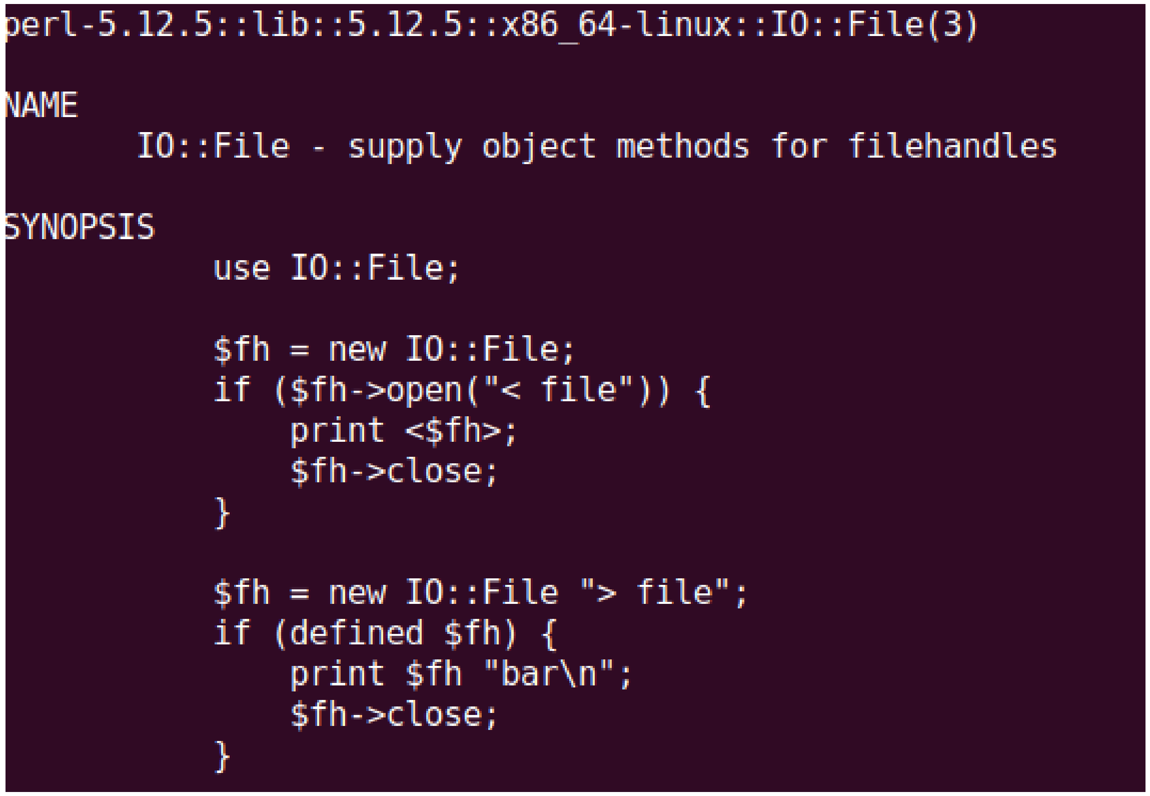
continued...

## New-Fangled Prototypes

- Core developers long recognized that vintage-1994 prototypes were obstacle to getting signatures right
- Introduced `:prototype($$)` syntax in `perl-5.22` (2015)
- But still ... outside of Catalyst, who really likes subroutine attributes?

## Indirect Object Syntax (1)

- Was indirect object syntax introduced only to keep `c++` programmers happy?
- Somewhat "discouraged" as early as 2000
- ... but present in documentation as late as `perl-5.12` (2010)



```
perl-5.12.5::lib::5.12.5::x86_64-linux::IO::File(3)

NAME
       IO::File - supply object methods for filehandles

SYNOPSIS
           use IO::File;

           $fh = new IO::File;
           if ($fh->open("< file")) {
               print <$fh>;
               $fh->close;
           }

           $fh = new IO::File "> file";
           if (defined $fh) {
               print $fh "bar\n";
               $fh->close;
           }
```

## The Perl 7 Project

- So now you’re probably wondering ...

continued...

## The Perl 7 Project

- So now you’re probably wondering ...
- How is Perl 7 actually being brought to life?

continued...

## Indirect Object Syntax (2)

- Still officially supported, but `no feature 'indirect';` is better practice
- Will not be upon-request-only until later in Perl 7 development cycle
  - "Breaks" a lot of CPAN ...
  - ... but the damage is easily repairable
- Not yet clear whether we'll warn about indirect object syntax in 7.0.0

## The Perl 7 Project

- So now you’re probably wondering ...

continued...

## The Perl 7 Project

- So now you’re probably wondering ...
- How is Perl 7 actually being brought to life?

continued...

## The Perl 7 Project

- So now you’re probably wondering ...
- How is Perl 7 actually being brought to life?
- `How can I help?`

## The Perl 7 Project

- So now you’re probably wondering ...

continued...

## The Perl 7 Project

- So now you’re probably wondering ...
- How is Perl 7 actually being brought to life?

continued...

## Development of Perl 7

- Perl 7 is not being developed in the main Perl repository on GitHub
- It is being developed in Nicolas Rochelemagne’s perl repository on GitHub

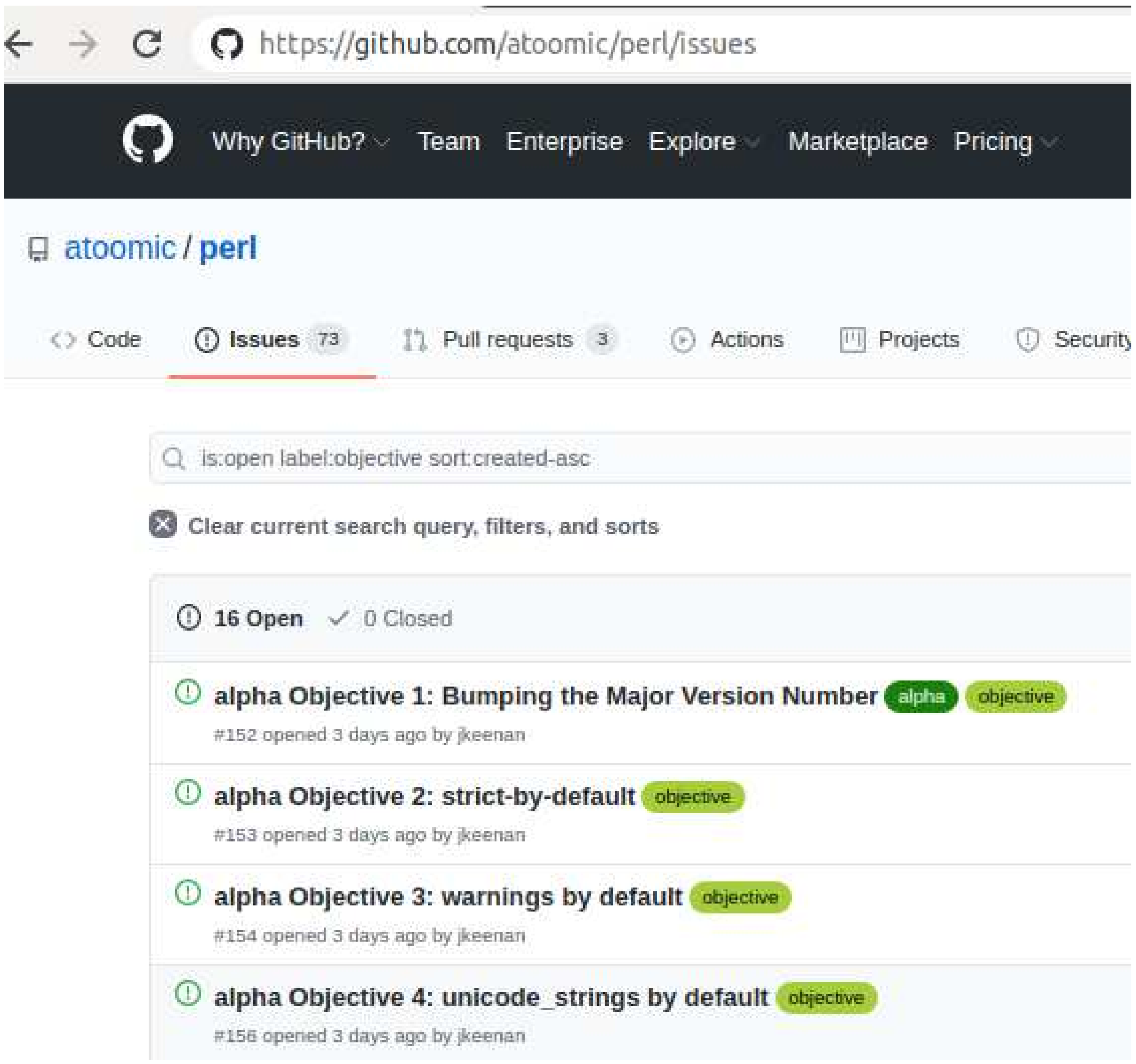
alpha branch in Nico’s repository

- Main branch there is now called `alpha`
- To avoid complaints from opponents, we have had to characterize this as a "research" effort
- ... but in practice there is no other implementation of Perl 7 being worked on

## Perl 7 Issue Tracker

- We track issues in atoomic’s repo as well

alpha Issue Tracker



https://github.com/atoomic/perl/issues

atoomic / perl

is:open label:objective sort:created-asc

Clear current search query, filters, and sorts

16 Open ✓ 0 Closed

alpha Objective 1: Bumping the Major Version Number alpha objective
#152 opened 3 days ago by jkeenan

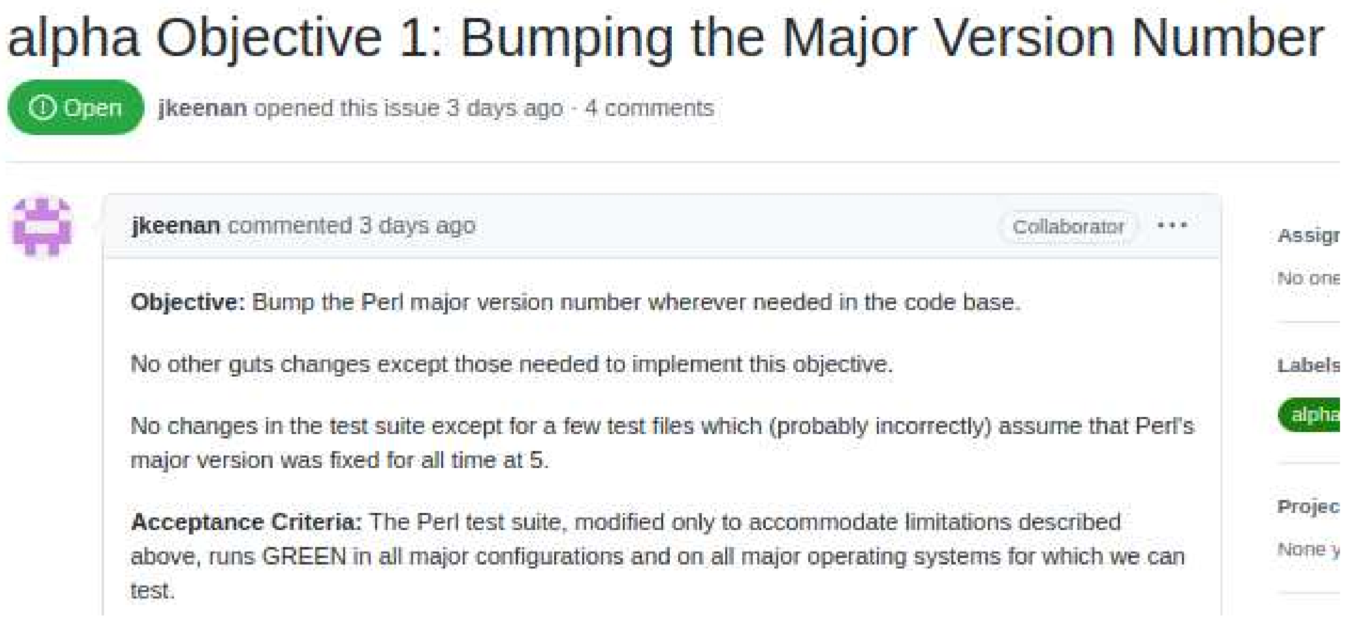
alpha Objective 2: strict-by-default objective
#153 opened 3 days ago by jkeenan

alpha Objective 3: warnings by default objective
#154 opened 3 days ago by jkeenan

alpha Objective 4: unicode_strings by default objective
#156 opened 3 days ago by jkeenan

## Perl 7 Development Objectives

- We've decided to develop Perl 7 through a series of clear, distinct development objectives
- Objective 1: Bumping the major version number



alpha Objective 1: Bumping the Major Version Number

Open jkeenan opened this issue 3 days ago · 4 comments

jkeenan commented 3 days ago Collaborator ···

**Objective:** Bump the Perl major version number wherever needed in the code base.

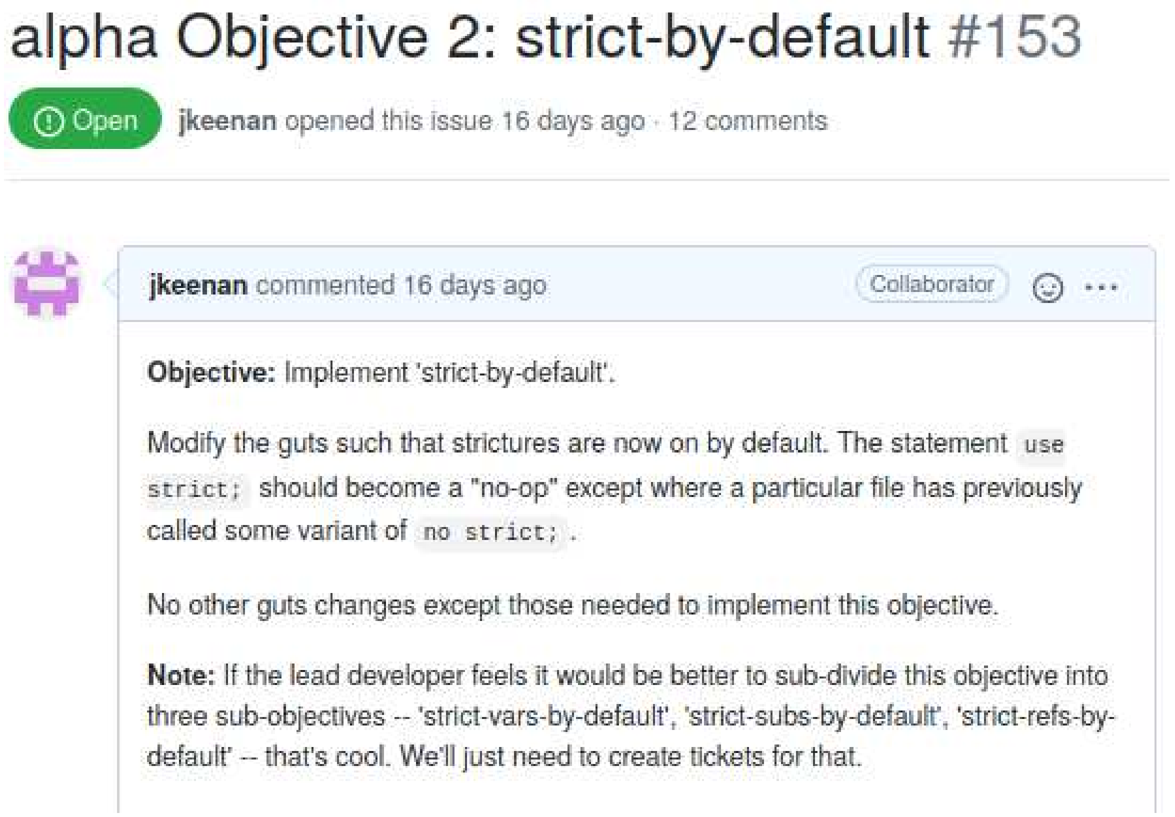
No other guts changes except those needed to implement this objective.

No changes in the test suite except for a few test files which (probably incorrectly) assume that Perl's major version was fixed for all time at 5.

**Acceptance Criteria:** The Perl test suite, modified only to accommodate limitations described above, runs GREEN in all major configurations and on all major operating systems for which we can test.

## Objective 2: Strict-by-Default

- We reached Objective 1 in early August began work on Objective 2



alpha Objective 2: strict-by-default #153

**jkeenan** opened this issue 16 days ago · 12 comments

**jkeenan** commented 16 days ago

**Objective:** Implement 'strict-by-default'.

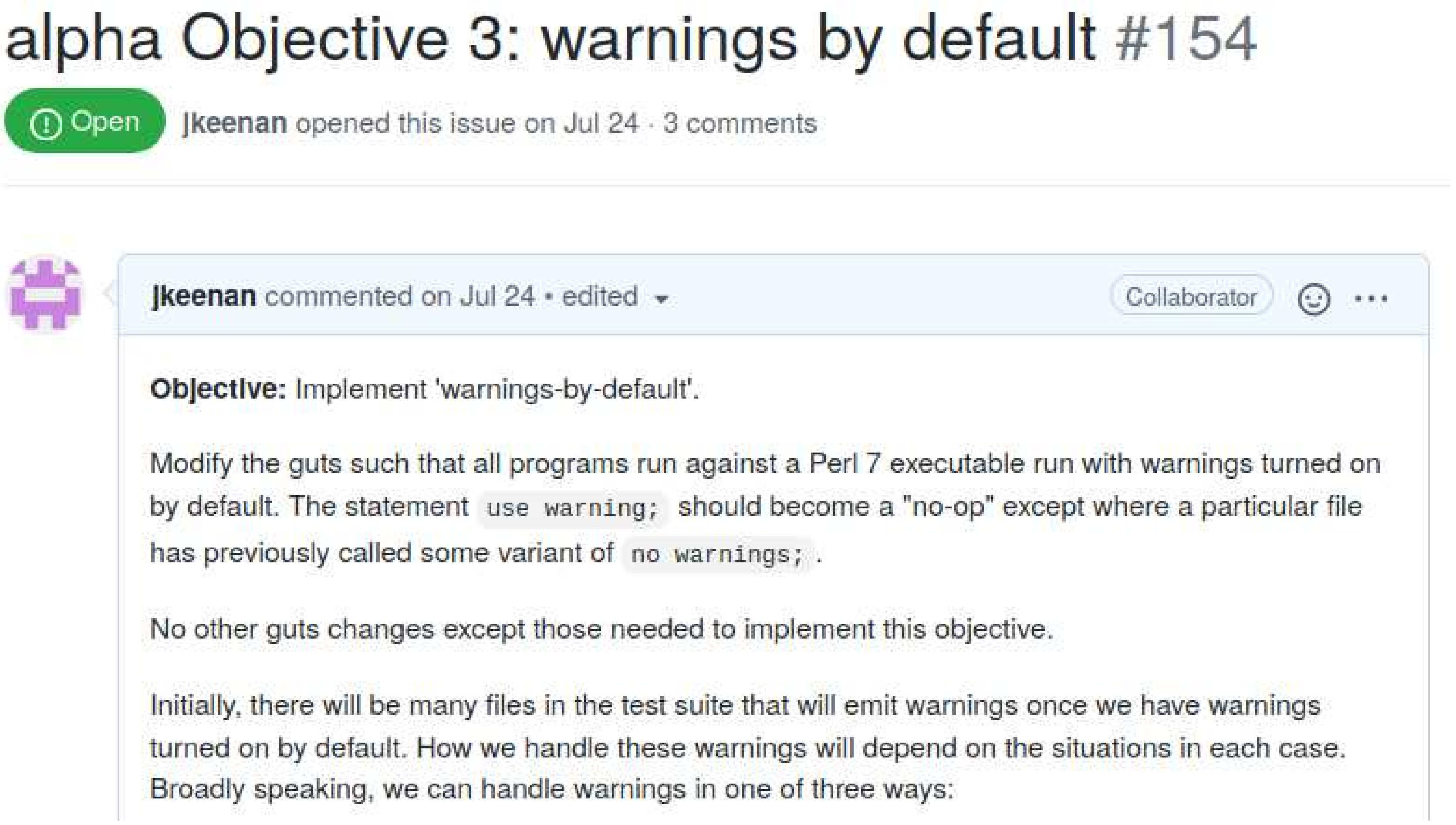
Modify the guts such that strictures are now on by default. The statement `use strict;` should become a "no-op" except where a particular file has previously called some variant of `no strict;`.

No other guts changes except those needed to implement this objective.

**Note:** If the lead developer feels it would be better to sub-divide this objective into three sub-objectives -- 'strict-vars-by-default', 'strict-subs-by-default', 'strict-refs-by-default' -- that's cool. We'll just need to create tickets for that.

## Objective 3: Warnings-by-Default

- We reached Objective 2 on September 20 and are now working on Objective 3



alpha Objective 3: warnings by default #154

jkeenan opened this issue on Jul 24 · 3 comments

jkeenan commented on Jul 24 • edited

**Objective:** Implement 'warnings-by-default'.

Modify the guts such that all programs run against a Perl 7 executable run with warnings turned on by default. The statement `use warning;` should become a "no-op" except where a particular file has previously called some variant of `no warnings;` .

No other guts changes except those needed to implement this objective.

Initially, there will be many files in the test suite that will emit warnings once we have warnings turned on by default. How we handle these warnings will depend on the situations in each case. Broadly speaking, we can handle warnings in one of three ways:

## (Smoke-)Test Driven Development

- In Perl 5, we place great importance on smoke-testing results we gather from combinations of:
  - Different operating systems
  - Different versions of those OSes
  - Different ways of configuring `perl` on those machines
- It's not sufficient for us to simply run `make test` on an unthreaded build on Linux, get all tests `PASSing` and call it a day
- Whenever we see new `FAILures` in a smoke-test report, we have to investigate.

## (Smoke-)Test Driven Development in Perl 7

- In Perl 7, we learned (the hard way) that we have to adopt this approach as well
- If we get 99% of our tests `PASSing` on an unthreaded build on Linux, but only 50% `PASSing` on a threaded build on FreeBSD, then we can't release

## What We’re Working on Right Now

- We have 15 objectives
- We’ve achieved Objective 1: bumping the major version number

alpha Objective 1: Bumping the Major Version Number

- We’ve achieved Objective 2: strict-by-default
  - There are 1000s of test files in the core distribution test suite
  - In our Perl 7 repository, they are now entirely `strict`-compliant

alpha Objective 2: strict-by-default

- Now working on Objective 3: warnings-by-default

alpha Objective 3: warnings-by-default

## How Can I Help?

- `You` can help by taking on an assignment to study a certain number of test files:
  - Not just "code reviewing" -- more "code understanding"
  - Understand the files as they appear in Perl 5 repository
  - Understand how they've been modified in Perl 7 repository for version bump and strict-by-default
  - Test them on all the platforms to which you have access; make sure they `PASS` on a variety of build configurations
    - unthreaded, threaded, non-debugging, debugging, etc.
- `We will train you in how to work with the Perl core distribution!`
- Contact: `jkeenan at pobox dot com`
- IRC: irc.perl.org #p7-dev (nick: kid51)

## Thanks!

## One Perl Executable to Rule Them All?

- Vendors of operating systems -- whether commercial or non-profit -- either:
  - Compile a `perl` executable which they install as `/usr/bin/perl`
    - "System perl"
  - Compile a `perl` executable which they offer as a "port" or "package"
    - "Vendor perl"
    - Gets installed in places like `/usr/local/bin/perl`

continued...

## Bonus Slides

## One Perl Executable to Rule Them All?

- Vendors of operating systems -- whether commercial or non-profit -- either:
    - Compile a `perl` executable which they install as `/usr/bin/perl`
        - "System perl"
    - Compile a `perl` executable which they offer as a "port" or "package"
        - "Vendor perl"
        - Gets installed in places like `/usr/local/bin/perl`

continued...

## Perl 7: What Are the Objections?

- Disaster!
- Dead on arrival!
- Will break CPAN!
- Will break backwards compatibility!
- Massive reputational damage!

## One Perl Executable to Rule Them All?

- Vendors of operating systems -- whether commercial or non-profit -- either:
    - Compile a `perl` executable which they install as `/usr/bin/perl`
        - "System perl"
    - Compile a `perl` executable which they offer as a "port" or "package"
        - "Vendor perl"
        - Gets installed in places like `/usr/local/bin/perl`

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
  - Not evolving, like ...

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
  - Not evolving, like ...
  - `python`

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
    - Not evolving, like ...
    - `python`
- "That `perl` one-liner I embedded in a `bash` script for Goldman Sachs back in 1996 has made millions -- so it damn well better keep running!"

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
  - Not evolving, like ...
  - `python`
- "That `perl` one-liner I embedded in a `bash` script for Goldman Sachs back in 1996 has made millions -- so it damn well better keep running!"
  - Did Goldman Sachs ever say thank-you to the Perl community for that functionality?

continued...

## One Perl Executable to Rule Them All?

- Vendors of operating systems -- whether commercial or non-profit -- either:
  - Compile a `perl` executable which they install as `/usr/bin/perl`
    - "System perl"
  - Compile a `perl` executable which they offer as a "port" or "package"
    - "Vendor perl"
    - Gets installed in places like `/usr/local/bin/perl`
- It’s convenient, since the end user does not have to configure, compile and install their own `perl`, but ...
- "The system `perl` is for the system" -- `brian.d.foy`

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
  - Not evolving, like ...

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
    - Not evolving, like ...
    - `python`

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
  - Not evolving, like ...
  - `python`
- "That `perl` one-liner I embedded in a `bash` script for Goldman Sachs back in 1996 has made millions -- so it damn well better keep running!"

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
  - Not evolving, like ...
  - `python`
- "That `perl` one-liner I embedded in a `bash` script for Goldman Sachs back in 1996 has made millions -- so it damn well better keep running!"
  - Did Goldman Sachs ever say thank-you to the Perl community for that functionality?

continued...

## Objectors Assume You Only Have One Perl Executable

- Yes, if some sysadmin were to change `/usr/bin/perl` to be Perl 7 rather than Perl 5, that would bork your programs
- But only if you were relying on `/usr/bin/perl` to execute your programs in the first place!
- That hasn't been "best practice" in this millennium
- Most applications from some kind of container: virtual machine, jail, docker
  - Compile your own `perl` and install it in the application's container

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
    - Not evolving, like ...

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
    - Not evolving, like ...
    - `python`

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
    - Not evolving, like ...
    - `python`
- "That `perl` one-liner I embedded in a `bash` script for Goldman Sachs back in 1996 has made millions -- so it damn well better keep running!"

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
  - Not evolving, like ...
  - `python`
- "That `perl` one-liner I embedded in a `bash` script for Goldman Sachs back in 1996 has made millions -- so it damn well better keep running!"
  - Did Goldman Sachs ever say thank-you to the Perl community for that functionality?

continued...

## "But It Breaks Backwards Compatibility -- Perl's Primary Asset"

- For some people, the fact that Perl's defaults have not changed since 1994 is the best thing about it
- They want `perl` to be stable like `cat`, `ls`, `sed` or `awk`
  - Not evolving, like ...
  - `python`
- "That `perl` one-liner I embedded in a `bash` script for Goldman Sachs back in 1996 has made millions -- so it damn well better keep running!"
  - Did Goldman Sachs ever say thank-you to the Perl community for that functionality?
    - Just askin'

## Interoperability: Is It Really Needed?

- Personally, I doubt it
    - Perl 7's best use will be in new development

continued...

## Interoperability: Is It Really Needed?

- Personally, I doubt it
  - Perl 7's best use will be in new development
- A major version release should `not promise 100% interoperability` with earlier major versions

continued...

## Interoperability: Is It Really Needed?

- Personally, I doubt it
  - Perl 7's best use will be in new development
- A major version release should `not promise 100% interoperability` with earlier major versions
- Instead, it should `provide significant interoperability`

## Interoperability: Objective 15

- I believe we should only implement interoperability between Perl 5 and Perl 7 once we have implemented all aspects of improved defaults (objectives 2 through 15)

continued...

## Interoperability: Objective 15

- I believe we should only implement interoperability between Perl 5 and Perl 7 once we have implemented all aspects of improved defaults (objectives 2 through 15)
- In particular, we should not switch to Perl 5 semantics within a test file just to get the test to PASS.
- If we do that, we're not really testing Perl 7 semantics

## Suppressing Warnings (1)

- We want to honor the original test file author’s intent as to what is being tested
- We suppress a warning if the warning is not pertinent to what we’re testing

```
$ cat foo.pl
use strict;
use warnings;
my ($i, $j, $k);
$i = 'foo';
$k = $i . $j;
print "$k\n";
```

continued...

## Suppressing Warnings (1)

- We want to honor the original test file author's intent as to what is being tested
- We suppress a warning if the warning is not pertinent to what we're testing

```
$ cat foo.pl
use strict;
use warnings;
my ($i, $j, $k);
$i = 'foo';
$k = $i . $j;
print "$k\n";

$ perl foo.pl
Use of uninitialized value $j in concatenation (.) or string at foo.pl line 5.
foo
```

## Suppressing Warnings (2)

- A "Perl 7-ish" program that assumes `strict` and `warnings` on by default

```
$ cat foo7.pl
my ($i, $j, $k);
$i = 'foo';
$k = $i . $j;
print "$k\n";
```

continued...

## Suppressing Warnings (2)

- A "Perl 7-ish" program that assumes `strict` and `warnings` on by default

```
$ cat foo7.pl
my ($i, $j, $k);
$i = 'foo';
$k = $i . $j;
print "$k\n";

$ p7 foo7.pl
Use of uninitialized value $j in concatenation (.) or string at foo7.pl line 3.
foo
```

## Suppressing Warnings (3)

- That "Perl 7-ish" program corrected

```
$ cat foo7a.pl
my ($i, $j, $k);
$i = 'foo';
{
    no warnings 'uninitialized';
    $k = $i . $j;
}
print "$k\n";
```

continued...

## Suppressing Warnings (3)

- That "Perl 7-ish" program corrected

```
$ cat foo7a.pl
my ($i, $j, $k);
$i = 'foo';
{
    no warnings 'uninitialized';
    $k = $i . $j;
}
print "$k\n";
```

- perl-5.32.0 designated maintenance-only; `no new features`

  $ `p7 foo7a.pl` foo

## Capturing and Testing Warnings (1)

- Sometimes a newly visible warning is significant ...
  - ... significant enough that we want to verify that it was emitted
- From: `t/uni/readline.t` in blead

```
use constant roref=>\2;
eval { for (roref) { $_ = <FÊ> } };
like($@, qr/Modification of a read-only value attempted/, '[perl #19566]');
```

## Eliminating Warnings (2)

- Note that file does not have `use warnings;`
- Run this file in Perl 5 (adapted from `t/op/groups.t`

```
$ perl heed-syntax-warning.t
ok 1 # skip uid!=0
1..1
```

continued...

## Eliminating Warnings (2)

- Note that file does not have `use warnings;`
- Run this file in Perl 5 (adapted from `t/op/groups.t`

```
$ perl heed-syntax-warning.t
ok 1 # skip uid!=0
1..1
```

- Now call for warnings in Perl 5

```
$ perl -w heed-syntax-warning.t
Scalar value @sup_group[2] better written as $sup_group[2] at heed-syntax-warning.t line 18.
ok 1 # skip uid!=0
1..1
```

continued...

## Capturing and Testing Warnings (2)

- From: `t/uni/readline.t` in core-p7 branch

```
use constant roref=>\2;
{
    # Suppress an unimportant warning
    no warnings 'once';
    # Capture important warnings
    my @these_warnings = ();
    local $SIG{__WARN__} = sub { push @these_warnings, $_[0]; };
    eval { for (roref) { $_ = <FÊ> } };
    like($@, qr/Modification of a read-only value attempted/, '[perl #19566]');
    # Test that we got the warnings we expected
    my $warnings_count = 0;
    for my $w (@these_warnings) {
        chomp $w;
        if ($w =~ m/^readline\(\) on unopened filehandle/) {
            $warnings_count++;
        }
    }
    is($warnings_count, 1,
        "Got expected number of 'unopened' warnings");
}
```

## Eliminating Warnings (2)

- Note that file does not have `use warnings;`
- Run this file in Perl 5 (adapted from `t/op/groups.t`

```
$ perl heed-syntax-warning.t
ok 1 # skip uid!=0
1..1
```

continued...

## Eliminating Warnings (2)

- Note that file does not have `use warnings;`
- Run this file in Perl 5 (adapted from `t/op/groups.t`

```
$ perl heed-syntax-warning.t
ok 1 # skip uid!=0
1..1
```

- Now call for warnings in Perl 5

```
$ perl -w heed-syntax-warning.t
Scalar value @sup_group[2] better written as $sup_group[2] at heed-syntax-warning.t line 18.
ok 1 # skip uid!=0
1..1
```

continued...

## Eliminating Warnings (1)

- Sometimes the warning needs to be taken seriously ...
  - ... meaning the code needs to be rewritten to eliminate the cause for the warning

```
$ cat heed-syntax-warning.t
use Test::More;

SKIP: {
    # try to add a group as supplementary group
    my $root_uid = 0;
    skip "uid!=0", 1 if $< != $root_uid and $> != $root_uid;
    my @groups = split ' ', $);
    my @sup_group;
    setgrent;
    while(my @ent = getgrent) {
        next if grep { $_ == $ent[2] } @groups;
        @sup_group = @ent;
        last;
    }
    endgrent;
    skip "No group found we could add as a supplementary group", 1
        if (!@sup_group);
    $) = "$) @sup_group[2]";
    my $ok = grep { $_ == $sup_group[2] } split ' ', $);
    ok $ok, "Group `$sup_group[0]' added as supplementary group";
}
```

```
        done_testing;
```

## Eliminating Warnings (2)

- Note that file does not have `use warnings;`
- Run this file in Perl 5 (adapted from `t/op/groups.t`

```
$ perl heed-syntax-warning.t
ok 1 # skip uid!=0
1..1
```

continued...

## Eliminating Warnings (2)

- Note that file does not have `use warnings;`
- Run this file in Perl 5 (adapted from `t/op/groups.t`

```
$ perl heed-syntax-warning.t
ok 1 # skip uid!=0
1..1
```

- Now call for warnings in Perl 5

```
$ perl -w heed-syntax-warning.t
Scalar value @sup_group[2] better written as $sup_group[2] at heed-syntax-warning.t line 18.
ok 1 # skip uid!=0
1..1
```

continued...

## Eliminating Warnings (2)

- Note that file does not have `use warnings;`
- Run this file in Perl 5 (adapted from `t/op/groups.t`

```
$ perl heed-syntax-warning.t
ok 1 # skip uid!=0
1..1
```

- Now call for warnings in Perl 5

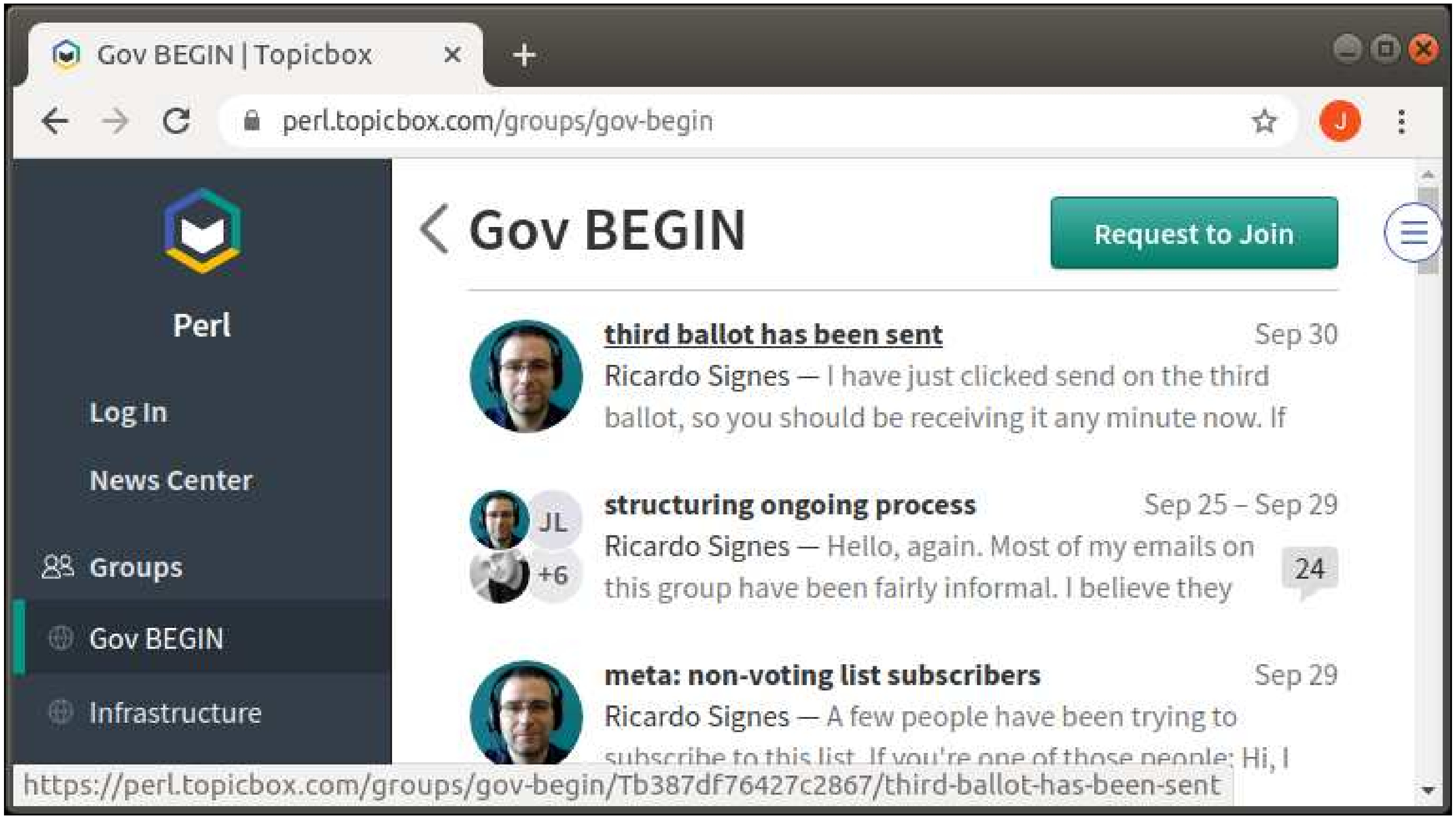
```
$ perl -w heed-syntax-warning.t
Scalar value @sup_group[2] better written as $sup_group[2] at heed-syntax-warning.t line 18.
ok 1 # skip uid!=0
1..1
```

- Here's the offending code

```
 16     skip "No group found we could add as a supplementary group", 1
 17         if (!@sup_group);
 18     $) = "$) @sup_group[2]";   # <---- Offender!
```

- Perl 7 won't let you ignore that!

## Governance Discussions



Gov BEGIN

**third ballot has been sent** Sep 30
Ricardo Signes — I have just clicked send on the third ballot, so you should be receiving it any minute now. If

**structuring ongoing process** Sep 25 – Sep 29
Ricardo Signes — Hello, again. Most of my emails on this group have been fairly informal. I believe they

**meta: non-voting list subscribers** Sep 29
Ricardo Signes — A few people have been trying to subscribe to this list. If you're one of those people: Hi, I

https://perl.topicbox.com/groups/gov-begin/Tb387df76427c2867/third-ballot-has-been-sent