Programming Leftovers

By Roy Schestowitz

Created 29/06/2020 - 6:29pm
Submitted by Roy Schestowitz on Monday 29th of June 2020 06:29:29 PM Filed under Development [1]

- **+1 "use v7;:" in Perl 7** [2]

  
  This syntax is the history of Perl and is also a mechanism for maintaining backward compatibility with newer versions of Perl.

  The reason this was not used is simply the small granularity.

  I don't remember much about Perl, so I can't tell the difference between use v5.20 and use v5.30.

  And because the warnings and utf8 aren't turn on, I couldn't find a meaning to actively use it.

  use v7; is very easy to understand.

  use v7;

  Imagine an application user actively uses it instead of writing use strict, use warnings, use utf8;,, use feature'say', ....

- **The [Perl] Weekly Challenge #066** [3]

  The much awaited event, **Conference in the Cloud?** took most of my free time. Having said that I still managed to do Live Coding YouTube videos for Divide Integers and Power Integers.

  I really enjoyed both tasks, specially Power Integers. It didn't take long to solve both tasks. I was able to get it done by midweek. But for YouTube video, I had to wait until the conference was over. Thanks to the Chief Editor of Perl Weekly newsletter editorial note, I now have 67 subscribers to my YouTube Channel. I would like to thank each and every subscriber. I promise to do regular video every week.
Using Bash traps in your scripts [4]

It's easy to detect when a shell script starts, but it's not always easy to know when it stops. A script might end normally, just as its author intends it to end, but it could also fail due to an unexpected fatal error. Sometimes it's beneficial to preserve the remnants of whatever was in progress when a script failed, and other times it's inconvenient. Either way, detecting the end of a script and reacting to it in some pre-calculated manner is why the Bash trap directive exists.

Oracle Helidon 2.0 reaches general availability [5]

Global enterprise database and software vendor, Oracle, has announced the general availability of Helidon 2.0, a set of Java libraries simplifying microservices development.

Some Open-Source Projects Begin Quickly Working Towards macOS ARM64 Support [6]

While the first MacBooks / Macs with Apple's 64-bit ARM chips won't be shipping to consumers until around the end of the year and Apple is only sending out a limited number of developer systems, some open-source projects have already been making the necessary build system changes and other preparations for 64-bit ARM Mac builds. This work can be started by untangling assumptions in some of these projects that when building for macOS/Darwin means x86_64 and in some cases better modularizing their logic where they support iOS already with similar chips to what will be appearing in these future computers. Changes can also be started around "fat" binaries for supporting macOS builds that support both x86_64 and ARM64/AArch64.

Building a startup using Crystal and Lucky [7]

Crystal and Lucky are not, in my opinion, ready for the inexperienced programmer. With over 40 years of programming experience, I have still faced challenges.

The power of Crystal's macro language means that it is used extensively in packages as powerful as the Lucky web platform. Unfortunately, this means that your programming errors are reported where they occur somewhere in a macro expansion, rather than where you have made them ? as you could expect were you calling into functions and methods rather than macros. The result is that error messages resulting from my use of Lucky are often simply indecipherable, yielding neither the location of their origin or, sometimes, even any information about the erroneous statement rather than some macro transformation of that
statement. Since the macro system is a code transformation machine, its arguments are not naturally as tightly typed as the rest of the Crystal language. Achieving good error reports for Lucky may require manually-added code to more tightly check the arguments to every macro. Fortunately, the macro mechanism does provide the framework to do such checking, AST nodes yield type information and the file name and line number of where they originate. I don’t know if there is anything that the compiler developers can do to improve error messages regarding macro expansions.

- **Towards greater ecological validity in security usability** [8]

When you are a medical doctor, friends and family invariably ask you about their aches and pains. When you are a computer specialist, they ask you to fix their computer. About ten years ago, most of the questions I was getting from friends and family as a security techie had to do with frustration over passwords. I observed that what techies had done to the rest of humanity was not just wrong but fundamentally unethical: asking people to do something impossible and then, if they got [cr]acked, blaming them for not doing it.

**Development**

**Source URL:** http://www.tuxmachines.org/node/139298

**Links:**