Programming Leftovers

By *Roy Schestowitz*
Created *22/07/2021 - 1:15am*
Submitted by Roy Schestowitz on Thursday 22nd of July 2021 01:15:15 AM Filed under Development [1]

- **The Rust Programming Language Blog: Rust 2021 public testing period [2]**

  We are happy to announce that the Rust 2021 edition is entering its public testing period. All of the planned features for the edition are now available on nightly builds along with migrations that should move your code from Rust 2018 to Rust 2021. If you'd like to learn more about the changes that are part of Rust 2021, check out the nightly version of the Edition Guide.

- **Released Giblog 2.0, and a movie "How to create your web site using Giblog and Perl" [3]**

  Released Giblog 2.0. GIblog is a tool to create your web site easily.

- **Ravgeet Dhillon: Progress Bar in Next.js [4]**

  Sometimes when we transition from one route to another, it takes a little time to do so due to different factors. Behind the scenes, it may be rendering a complex page component or doing an API call. In such cases, the app looks like it has frozen for some seconds and then suddenly transitions to the next route. This results in a poor UX. In such cases, it is better to add a progress bar to our application which gives our users a sense that something is loading.

-

Delivering Common Lisp executables using Consfigurator [5]

I realised this week that my recent efforts to improve how Consfigurator makes the fork(2) system call have also created a way to install executables to remote systems which will execute arbitrary Common Lisp code. Distributing precompiled programs using free software implementations of the Common Lisp standard tends to be more of a hassle than with a lot of other high level programming languages. Executables will often be hundreds of megabytes in size even if your codebase is just a few megabytes, because the whole interactive Common Lisp environment gets bundled along with your program?s code. Commercial Common Lisp implementations manage to do better, as I understand it, by knowing how to shake out unused code paths. Consfigurator?s new mechanism uploads only changed source code, which might only be kilobytes in size, and updates the executable on the remote system. So it should be useful for deploying Common Lisp-powered web services, and the like.

Here?s how it works. When you use Consfigurator you define an ASDF system ? analagous to a Python package or Perl distribution ? called your ?consfig?. This defines HOST objects to represent the machines that you?ll use Consfigurator to manage, and any custom properties, functions those properties call, etc.. An ASDF system can depend upon other systems; for example, every consfig depends upon Consfigurator itself. When you execute Consfigurator deployments, Consfigurator uploads the source code of any ASDF systems that have changed since you last deployed this host, starts up Lisp on the remote machine, and loads up all the systems. Now the remote Lisp image is in a similarly clean state to when you?ve just started up Lisp on your laptop and loaded up the libraries you?re going to use. Only then are the actual deployment instructions are sent on stdin.

- **Write your first web component** [6]

  Web components are a collection of open source technologies such as JavaScript and HTML that allow you to create custom elements that you can use and reuse in web apps. The components you create are independent of the rest of your code, so they're easy to reuse across many projects.

- **Josef Strzibny: Elixir Authorization Plugs** [7]

  Similar to Ruby?s Rack, Plug is a general specification for composing modules between web applications and application servers. Here?s how we can use them to build authorized pipelines in your router.

  Note that this post is not about whether you should do authorization at the router level. It?s likely you?ll do it as part of your business logic for the most part. But when it makes sense, you can use Plugs.

- **AMD AOCC 3.1 Compiler Released - Rebased On LLVM 12.0 [8]**

  AMD earlier this week quietly published a new version of its AOCC code compiler that is now rebased against the upstream LLVM/Clang 12.0 compiler state.

  AMD Optimizing C/C++ Compiler 3.0 was released back in March alongside the EPYC 7003 "Milan" launch. AOCC 3.1 is now available as the latest incremental improvement to this LLVM/Clang downstream that focuses on carrying various out-of-tree patches optimizing the open-source compiler for AMD's Zen microarchitecture family, making Flang suitable for compiling more Fortran code-bases, and other enhancements when building code for AMD CPUs.

Development

Source URL: http://www.tuxmachines.org/node/153669

Links:
[1] http://www.tuxmachines.org/taxonomy/term/145
[2] https://blog.rust-lang.org/2021/07/21/Rust-2021-public-testing.html
[3] http://blogs.perl.org/users/yuki_kimoto/2021/07/released-giblog-20-and-a-movie-how-to-create-your-web-site-using-giblog-and-perl.html
[4] https://www.ravsam.in/blog/progress-bar-in-next-js/
[5] https://spwhitton.name//blog/entry/delivering-lisp-images/
[6] https://opensource.com/article/21/7/web-components
[7] https://nts.strzibny.name/elixir-authorization-plugs/
[8] https://www.phoronix.com/scan.php?page=news_item&px=AMD-AOCC-3.1-Released