Programming Leftovers

By *Roy Schestowitz*
Created *19/09/2021 - 12:16pm*
Submitted by Roy Schestowitz on Sunday 19th of September 2021 12:16:16 PM Filed under Development [1]

- **Python in 2021: The Good, The Bad, and the Ugly [2]**

  In this post, I want to look at the biggest strengths and weaknesses of Python, with more emphasis on the weaknesses, just because these problems have been there for years now and some of the rough edges bleed a lot.

- **GitLab files to go public as both revenue and losses surge [3]**

  GitLab Inc., which provides a cloud service to enable software developers to share code and collaborate on projects, today announced plans to go public with an initial offering of stock.

  The San Francisco-based company, which counts among its competitors Microsoft Corp.-owned GitHub and Atlassian Corp. PLC?s BitBucket, didn?t reveal yet how much it plans to raise or precisely when it will do the IPO. It was last valued at $6 billion after a secondary share sale in January, and has raised a total of $400 million from investors such as Khosla Ventures, Altimeter Capital, TCV, Franklin Templeton and Coatue Management.

- **[Old] LLVM internals, part 1: the bitcode format [4]**

  I?ve done a couple of posts on LLVM itself, mostly on things you can do with LLVM or how LLVM represents particular program features.

  I?ve received some good feedback on these, but I?d like to focus a sub-series of posts on LLVM?s implementation itself: the file formats, parsing strategies, and algorithmic choices underlying LLVM?s public interfaces (APIs, CLIs, and consumable output files). I?ll be

writing these posts as I work on a series of pure-Rust libraries for ingesting LLVM?s intermediate representation, with the end goal of being able to perform read-only analyses of programs compiled to LLVM IR in pure Rust.

For those who don?t know what LLVM is, this post has a broader background on LLVM?s components and intermediate representation.

- **[Old] LLVM internals, part 2: parsing the bitstream [5]**

  In the last post, I performed a high-level overview of LLVM?s bitcode format (and underlying bitstream container representation). The end result of that post was a release announcement for llvm-bitcursor, which provides the critical first abstraction(s) for a pure-Rust bitcode parser.

  This post will be a more concrete walkthrough of the process of parsing actual LLVM bitstreams, motivated by another release announcement: llvm-bitstream.

  Put together, the llvm-bitcursor and llvm-bitstream crates get us two thirds-ish of the way to a pure-Rust parser for LLVM IR. The only remaining major component is a ?mapper? from the block and record representations in the bitstream to actual IR-level representations (corresponding to llvm::Module, llvm::Function, &c in the official C++ API).

- **LLVM internals, part 3: from bitcode to IR [6]**

  This post marks a turning point: now that we have reasonable abstractions for the bitstream container itself, we can focus on mapping it into a form that resembles LLVM?s IR. We?ll cover some of the critical steps in that process1 below, introducing a new crate (llvm-mapper) in the process.

  Also, critically: this post is the first in the series where our approach to parsing and interpreting LLVM?s bitcode differs significantly from how LLVM?s own codebase does things. The details of the post should still be interesting to anyone who wants to learn the details of how an IR-level LLVM module is constructed, but will not reflect how LLVM itself does that construction2.

Development

Source URL: http://www.tuxmachines.org/node/155808

Links:
[1] http://www.tuxmachines.org/taxonomy/term/145
[2] https://new.pythonforengineers.com/blog/python-in-2021-the-good-the-bad-and-the-ugly/
[3] https://siliconangle.com/2021/09/17/gitlab-files-go-public-revenue-losses-surge/
[4] https://blog.yossarian.net/2021/07/19/LLVM-internals-part-1-bitcode-format

[5] https://blog.yossarian.net/2021/08/10/LLVM-internals-part-2-parsing-the-bitstream
[6] https://blog.yossarian.net/2021/09/14/LLVM-internals-part-3-from-bitcode-to-IR