Scott Olson

scott@solson.me https://solson.me https://github.com/tsion

Experience

IBM Sept 2014–Aug 2015
Software Engineering Intern Markham, ON

- Began the porting effort of Clang to z/OS for IBM's XL C/C++ compiler
  - Involved low-level platform support and an understanding of UNIX and POSIX
- Diagnosed bugs in the compiler we were using to compile the Clang sources
  - Involved complex details of C++ features and an understanding of standards compliance
- Worked with C, C++, Makefiles, shell scripts, Python, and autoconf

Google May–Aug 2014
Software Engineering Intern Waterloo, ON

- Developed internal web tools for monitoring ads infrastructure
- Worked with C++, JS, SQL, HTML, and CSS
- Evolved tools based on feedback and meetings with actual users

Human-Computer Interaction Lab [link] May–Dec 2013, Sept 2015–Present
Research Assistant Saskatoon, SK

- Developed research projects from scratch and worked on existing projects
- Used C#, Java (Android), and some C++
- Guided research participants through user studies

Education

Bachelor of Science, Computer Science (Honours) 2011–2016
University of Saskatchewan Saskatoon, SK

- Arts & Science Dean's List (2011–2014)
- Internship during May 2014–August 2015 (no classes)
- Expected date of completion: May 2016

Skills

Languages

- Expert in C++, C, Rust, and Ruby
- I learn programming languages and write compilers as a hobby and have a deep knowledge of programming language design and implementation
- Familiar with Haskell, x86 assembly, JS, HTML, CSS, LaTeX, Idris, Coq, C#, Java, Python, Clojure, Lua, Matlab, Racket, SML, Forth, Nix, and have played with many others

Tools

- Quite familiar with LLVM (some of my hobby compilers used LLVM, my IBM work involved LLVM as a Clang dependency, and it comes up a lot in Rust since rustc uses an LLVM back-end)
- Regularly use git, vim, ssh, tmux, mosh, bash, fish, and gdb
- Previously used Arch Linux exclusively, now using Ubuntu and NixOS

PROJECTS [link]

Miri (Rust · 2015–2016)

- My undergraduate honours research project
- An experimental interpreter for the Rust compiler's mid-level intermediate representation (MIR)
- Different from most interpreters since it simulates differently-sized values, a virtual memory system, and unsafe memory operations (where invalid operations are detected and diagnosed)
- Based on a proposal for compile-time expression evaluation (like `constexpr` in C++) by one of the Rust compiler team members, who has helped me throughout this project

rustc (Rust · 2015–2016)

- Lately I've been contributing to the open-source Rust compiler partly for my work on Miri but mainly because I am fascinated by Rust
- I have learned a lot about compiler internals between rustc and Miri

Apricot (Ruby · 2012–2013)

- One of my more successful personal language projects
- Compiled a simple Clojure-like language to Rubinius bytecode
  - Rubinius is an alternative Ruby implementation with a C++ bytecode VM
  - Apricot code was able to call Ruby code and be called by Ruby code
- Collaborated remotely with a friend