Raymond Chang

raymondkenchang@gmail.com rkchang rkhchang raymondchang.ca

Skills

- Programming Languages: Python, C++, C, Java, SQL, TypeScript
- Technologies: Git, SQLite, GNU Make, Spring, CMake, PostgreSQL, Flask, GDB, Node.js, Vim
- Selected course work: Design & Analysis of Algorithms, Performance Engineering, Information Retrieval, Compiler Construction, Operating Systems, Databases, Artificial Intelligence, Language Design and Implementation

Education

University of Waterloo 09/2022 to 07/2024

Masters of Mathematics in Computer Science (Thesis) (95%)

Teaching Assistant for: Databases, Operating Systems, Algorithms, Software Analysis and Testing

University of Ottawa 09/2016 to 04/2021

Honours Bachelor of Science in Computer Science (CO-OP) (8.35/10) (Deans Honours List)

Honours project - *Modelling and verifying distributed leader election algorithms with TLA+*

Work Experience

Software Engineer Burnaby, Canada — 08/2024 to Present

Arista Networks

- Worked on a Python service that provides authentication and authorization capabilities for a network switch.
- Developing a clang-tidy check that parses C++ and inserts missing tracing calls.
- Patched a Syslog forwarding service written in C to enable additional configuration.
- Debugged race conditions in network switch authentication tests.

Research Assistant Waterloo, Canada — 09/2022 to 07/2024

University of Waterloo

- Compiler performance analysis and testing research

CoreOS Software Development Student Ottawa, Canada — 06/2020 to 08/2020

Blackberry QNX

- Developed a binary analysis tool in Python to scan and detect issues in the QNX kernel
- Wrote Unit tests in C for the QNX Neutrino kernel and hypervisor, boosting system reliability
- Used Git, GNU Make, Ghidra (reverse engineering tool)

CoreOS Software Development Student Ottawa, Canada — 09/2019 to 12/2019

Blackberry QNX

- Worked on a Python program to test and track new commits to Review Board thus streamlining the code review process
- Setup a QNX Hypervisor System with an Ubuntu guest to demonstrate its abilities to other teams
- Wrote Unit tests in C for the QNX Neutrino kernel and hypervisor, boosting system reliability

Telematics Control Unit Software Developer Ottawa, Canada — 01/2019 to 04/2019

Ford Motor Company

- Developed features and Unit tests for C and C++ multi-threaded Linux applications
- Fixed bugs and race conditions/deadlocks found by analysis tools such as Thread Sanitizer and Clang Static Analyzer
- Used GNU Make, Git, Jenkins and the Google repo tool

Junior Programmer Ottawa, Canada — 05/2018 to 08/2018

Canadian Border Services Agency

- Worked in a team of three to create a web application for the management of software development using Java, JSP, Hibernate, jQuery, SQL and Spring MVC
- Developed functional UI Mock-ups for the web application using HTML, CSS and JavaScript
- Gathered requirements for the web application and participated in weekly progress meetings
- Improved Unit test coverage of a seperate reporting application
- Acquired the Enhanced Reliability Status level of security clearance

Open Source Contributions

The LLVM Compiler Infrastructure Project https://llvm.org/

- Wrote C++ to emit an error when an unsupported OpenMP pragma is provided instead of crashing
- Added a unit test to validate my changes
- Merged Pull request: https://github.com/llvm/llvm-project/pull/70233

Selected Projects

Juice: Java Compiler - Course project for Compiler Construction 01/2023 to 04/2023

- Developed a Java compiler targeting x86 with two teammates
- Consisted of 28,753 lines of TypeScript, 628 Jest unit tests and passed the majority of test cases
- Ported and upgraded an existing Intermediate Representation (IR) interpreter from Java to TypeScript to improve the debugging process
- Wrote hundreds of Jest unit tests with final code coverage of 80%

Lettuce - New Programming Language and LLVM based compiler 07/2023 to 08/2023

- Developed new programming language and compiler in C++ using the LLVM compiler infrastructure
- Created a new Intermediate Representation using MLIR
- Source code: https://github.com/rkchang/mlidk