Louis Jenkins

34 W Montgomery Ave, Ardmore, Pennsylvania 19003, USA
LouisJenkinsCS@hotmail.com • +1 (610) 931-1207
https://www.linkedin.com/in/LouisJenkinsCS • http://github.com/LouisJenkinsCS • http://LouisJenkinsCS.github.io

EDUCATION

Bloomsburg University of Pennsylvania, Bloomsburg, PA

Bachelor of Science (B.S.) in Computer Science — May 2012 – May 2017

- Dean's List, GPA 3.0 — Spring 2014, Fall 2015, Fall 2016

RESEARCH EXPERIENCE

STUDENT RESEARCHER, Lehigh University, Bethlehem, PA — May 2016 – Jul 2016

- **Project:** Concurrent and Scalable Built-in Hash Table for the Go Programming Language (goo.gl/gclU8Z)
- **Advisor:** Michael F. Spear
- **Awards:**
  - Peer's Choice for Outstanding Project.
  - 1 of 2 nominees from non-PhD granting institutions to receive an Honorable Mention for the Computing Research Association's 2017 Outstanding Undergraduate Researchers, sponsored by **Microsoft Research**.
- **Publication:** L. Jenkins, T. Zhou, and M. Spear, "Concurrent Map for Go", in Submission to PACT 2017.
- **Summary:**
  - Designed and implemented a novel scalable lock-based concurrent map for Go's runtime and compiler.
  - Designed with compatibility with Go map syntax, supporitng insert/lookup/removal and concurrent iteration.
  - Outperforms sequential map by up to 7x across diverse microbenchmarks, competitive against lock-free maps.

MISC. EXPERIENCE

INDEPENDENT STUDY, Bloomsburg University, Bloomsburg, PA — Fall 2016

- **Project:** Open Source Software for Efficient Evaluation of Student Code (goo.gl/vpljef)
- **Advisor:** Drue Coles
- **Summary:**
  - Developed a free open source tool that helps automate the process of grading and leaving feedback for students.
  - Designed to promote a Write-Once Reuse-Anywhere philosophy of templated markups.
  - Implemented support for 169 languages and can be run on any platform with Java 8.

SKILLS

LANGUAGES

- **Proficient**: C, Java, Go
- **Familiar**: Haskell, C++, HTML/CSS/JavaScript

PERSONAL PROJECTS

MOLTAR-OS - HOBBY OPERATING SYSTEM (goo.gl/o27yoy)

- Developing an operating system in C and Assembly for the 32-bit x86 architecture for academic purposes.
  - Implemented virtual memory, interrupts, basic VGA and keyboard driver, and uniprocessor multitasking.
  - Designed to take a higher-half approach to virtual memory with 4MB pages.
  - Building from the ground up with POSIX-compliance as a long-term goal.
- Planned to have process management, multiprocessor support, ELF Binary loading, and command-line interface.
  - Planned to run actual userspace programs in a command-line environment.

MINIMAL-JVM - BYTECODE INTERPRETER (goo.gl/ewQHLW)

- Developed a java virtual machine in Haskell to run very simple programs written for JVM for academic purposes.
  - Implemented support for load/store, push/pop, conditional jumps, and arithmetic instructions.
  - Loads a single '.class' files that adhere to a subset of the JVM 8 Specification.
  - Designed execution cycle to be as functionally pure as possible through monads.
- Planned to have dynamic classloading, SSA-conversion of bytecode, and full support for bytecode instructions.
  - Planned experiment to test if functionally pure code under a functionally pure JVM is as efficient as others.

ANDROID WINDOW MANAGER (goo.gl/tYiOum)

- Created new window manager for Android, which allows the user to configure predefined Widgets.
  - Allows user to drag, resize, minimize, maximize, and snap Widgets via touch.
  - Handles marshalling and unmarshalling to maintain state across multiple user sessions.
  - Widgets include a web browser, notepad, screen recorder, and Google maps.
- In-progress features include support to allow user creation of custom Widgets using a built-in WYSIWYG editor.
  - Designing minimal interpreted language for user to create callbacks for certain actions.

UTILITIES PACKAGE FOR THE C PROGRAMMING LANGUAGE (goo.gl/wsz9wn)

- Created a large collection of useful thread-safe data structures and threading abstractions.
  - Includes Linked Lists, Blocking Queues, Hash Maps, and Lock-Free Queues and Stacks.
  - Incorporated some support for generic iteration through opaque, data-structure specific callbacks.
  - Developed threading abstractions, including a thread pool, events, and futures.
- Implemented memory reclaimation tools for sane memory management outside of a Garbage Collected environment.
  - Developed Hazard Pointer implementation for lock-free data-structures and Reference Counting.