Lab 9: Code Checking

Nathan Jarus

March 21, 2016

Introduction

This lab will give you experience using both Valgrind and Clang's code checking tools. Clone the repo down; don't forget to `source examples/symbolizer.sh`! (You may want to borrow the example makefile as well.)

Problem 1: Drip, Drip, Drip, Drip

1. Take a look at `prob1.cpp`. It's the linked list problem from last time, with the bugs fixed and a new one introduced.
2. Run it in valgrind. What is the problem?
3. Why is this problem occurring? (What memory is being allocated, and what memory is being freed?)
4. Fix the bug and check that your fix is correct.

Problem 2: Imagine the possibilities!

1. Take a look at `prob2.cpp`.
2. Without running the code: What section of the leak summary do you think Valgrind will report each block in?
3. Run the code through Valgrind. Were you right? If not, how did things differ?
4. Why do you think there is a difference between the `plain` and `dtor` classes? (Hint: C++ needs to know how many destructors to run when a block is freed.)

Problem 3: Use wisely

1. Take a look at `prob3.cpp`.
2. Build `prob3.cpp` with Clang's address sanitizer.
3. Run it. What is the result?
4. What is the problem?
5. Fix the bug and check that your fix is correct.