Labs for Computer Systems: A Programmer's Perspective (CS:APP)

Randal E. Bryant and *David R. O'Hallaron*

Instructor Site

Conditions of use: If you are an instructor with a CS:APP account, then you have permission to download, inspect, and try out these labs. However, you may use them in courses only if you have adopted the CS:APP textbook.

This page contains a complete set of turnkey labs for the CS:APP text. The labs all share some common features, described here.

Each lab is distributed in a self-contained tar file and has been compiled with GCC 2.95.3 and tested on a Linux/IA32 machine. You will need a CS:APP account to download the code. To untar foo.tar, type "tar xvf foo.tar" to the Unix shell. This will create a directory called "foo" that contains all of the material for the lab.

Labs that were updated after Sep 1, 2006 are marked as NEW .

- *Data Lab* *[Updated Aug 29, 2003]* (View the writeup) (Release Notes)

    Students implement simple logical and arithmetic functions, but using a highly restricted subset of C. For example, they must compute the absolute value of a number using only bit-level operations. This lab helps students understand the bit-level representations of C data types and the bit-level behavior of the operations on data.

- *Bomb Lab* *[Updated Mar 12, 2003]* (View the writeup) (Release Notes)

    A "binary bomb" is a program provided to students as an object code file. When run, it prompts the user to type in 6 different strings. If any of these is incorrect, the bomb ``explodes,'' printing an error message and logging the event on a grading server. Students must ``defuse'' their own unique bomb by disassembling and reverse engineering the program to determine what the 6 strings should be. The lab teaches students to understand assembly language, and also forces them to learn how to use a debugger. It's also great fun. A legendary lab among the CMU undergrads.

    Here's a Linux/IA32 binary bomb that you can try out for yourself. To download using either IE or Netscape, left click and save to a file called bomb.tar. Then copy bomb.tar to a Linux machine and expand into the binary executable bomb by typing "tar xvf bomb.tar". The feature that notifies the grading server has been disabled, so feel free to explode this bomb with impunity. If you're an instructor with a CS:APP account, then you can download the solution.

- *Buffer Lab* *[Updated Apr 28, 2004]* (View the writeup) (Release Notes)

Students modify the run-time behavior of a binary executable by exploiting a buffer overflow bug. This lab teaches the students about the stack discipline and teaches them about the danger of writing code that is vulnerable to buffer overflow attacks.

- *Architecture Lab* *[Updated Nov 4, 2004]* (View the writeup) (Release Notes)

  Students are given a small default Y86 array copying function and a working pipelined Y86 processor design that runs the copy function in some nominal number of clock cycles per array element (CPE). The students attempt to minimize the CPE by modifying both the function and the processor design. This gives the students a deep appreciation for the interactions between hardware and software.

  Note: The lab materials include the master source distribution of the Y86 processor simulators and the *Y86 Guide to Simulators*.

- *Performance Lab* *[Updated Jun 26, 2003]* (View the writeup) (Release Notes)

  Students optimize the performance of an application kernel function such as convolution or matrix transposition. This lab provides a clear demonstration of the properties of cache memories and gives them experience with low-level program optimization.

- *Shell Lab* *[Updated July 28, 2003]* (View the writeup) (Release Notes)

  Students implement their own simple Unix shell program with job control, including the `ctrl-c` and `ctrl-z` keystrokes, `fg`, `bg`, and `jobs` commands. This is the students' first introduction to application level concurrency, and gives them a clear idea of Unix process control, signals, and signal handling.

- *Malloc Lab* *[Updated April 28, 2004]* (View the writeup) (Release Notes)

  Students implement their own versions of `malloc`, `free`, and `realloc`. This lab gives students a clear understanding of data layout and organization, and requires them to evaluate different trade-offs between space and time efficiency. One of our favorite labs. When students finish this one, they really understand pointers!

- *Proxy Lab* *[Updated Mar 26, 2003]* (View the writeup) (Release Notes)

  Students implement a concurrent Web proxy that sits between their browser and the rest of the World Wide Web. This lab exposes the students to the interesting world of network programming, and ties together many of the concepts from the course, such as byte ordering, process control, signals, signal handling, file I/O, concurrency, and synchronization.

Questions or problems with your account? Forget your password? Send mail to Dave O'Hallaron

                24-11-2009 09:40