CSE 303: Concepts and Tools for Software Development

Dan Grossman

Spring 2007

Lecture 28— Code-Pointers vs. Objects; Course Wrap-Up

Where are we

- Final is Tuesday; today's material not on it
  - Emphasize second half of course, but Makefiles and being able to read basic Bash scripts is fair game
- Course evaluations for next 20 minutes
- Then one last "tie things together" piece and a few minutes of summary

You have grading to do

I am going to distribute course evaluation forms so you may rate the quality of this course. Your participation is voluntary, and you may omit specific items if you wish. To ensure confidentiality, do not write your name on the forms. There is a possibility your handwriting on the yellow written comment sheet will be recognizable; however, I will not see the results of this evaluation until after the quarter is over and you have received your grades. Please be sure to use a No. 2 PENCIL ONLY on the scannable form.

I have chosen *(name)* to distribute and collect the forms. When you are finished, he/she will collect the forms, put them into an envelope and mail them to the Office of Educational Assessment. If there are no questions, I will leave the room and not return until all the questionnaires have been finished and collected. Thank you for your participation.

I'll come back at 2:50.

Objects vs. Code Pointers

If you really study the provided code with this lecture I predict:

- You'll understand C, struct layouts, & code pointers much better
- You'll understand a lot about how objects are actually implemented
  - Which is one useful way to understand what they are

Key points:

- A pointer to an object can be subclassed with new private fields, making the `foreach` method in C++ *much* more useful than the `foreach` function in C
- Ways to give the C version the same usefulness:
  - "pass a `void*` that's passed back" (cf. `pthread_create`)
  - Code up OOP in C manually (passing "this", upcasting, etc.)

Course Summary

A lot of comfort and de-magicalizing

- 90% of you thought 90% of your classmates knew more of "that practical stuff"

A lot of pragmatic application of computer science

1. There are concepts behind `gdb`, but there's also value in knowing a couple handfuls of commands
2. The best computer scientists are not the folks who have the largest number of utilities, options, and features memorized
   - But the best computer scientists also don't know the least
   - And they're definitely not afraid to learn new ones

Some concepts too

We did *not* just pick up one strange set of rules after another!

- Shell-scripting is about automating program-invocation
- C programming is about a lower level of computing where:
  - all data is just bits and pointers are explicit
  - memory is managed manually
- Debuggers and profilers are about how to characterize and measure running programs
- Linking, make, etc. are about how code is created and combined to make running programs
- Assertions and specifications are about assumptions and the robustness of larger programs

Societal implications

And we talked about ethics too

- We talked about it like computer scientists
- (Overly) logical, rational, understanding limitations of technology and humans
- "We" should not be making all the decisions ourselves, but we should be actively engaged in the process

The Ultimate Goal

5 years from now, look at the course webpage and think this whole course was a waste of time.

After all, you won't remember not knowing this stuff.

And you won't understand why you need a teacher to help you pick up a new tool or automate a repetitive task.

From lecture 1: *There is an amorphous set of things computer scientists know about and novice programmers don't. Knowing them empowers you in computing, lessens the "friction" of learning in other classes, and makes you a mature programmer.*
"There is more to programming than Java methods"
"There is more to software development than programming"
"There is more to computer science than software development"
"There is more to computing's effects than computer science"