IN1: Evangelize Compilers

Why you should take a Compilers course

CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

Rich Programmer Food, Steve Yegge (Google)
http://steve-yegge.blogspot.ca/2007/06/rich-programmer-food.html

"If you don't know how compilers work, then you don't know how computers work."

"If you're not 100% sure whether you know how compilers work, then you don't know how they work."

Rich Programmer Food, Steve Yegge

"In fact, Compiler Construction is, in my own humble and probably embarrassingly wrong opinion, the second most important CS class you can take in an undergraduate computer science program."

Rich Programmer Food, Steve Yegge

“I'm not saying other CS courses aren't important, incidentally. Operating Systems, Machine Learning, Distributed Computing and Algorithm Design are all arguably just as important as Compiler Construction. Except that you can take them all and still not know how computers work, which to me means that Compilers really needs to be a mandatory 300-level course. But it has so many prerequisites that you can't realistically make that happen at most schools.”

Rich Programmer Food, Steve Yegge

- Situation 1: How do you configure your editor to auto-format a huge Java library according to your company's explicit and non-negotiable guidelines?
  - Steve's answer: You lobby your company to change the style guide to match whatever Eclipse does by default
  - Steve's real answer: Lexing and parsing

Rich Programmer Food, Steve Yegge

- Situation 2: Your company decides to do automatic documentation extraction from Javascript code. How do you write your own jsdoc extractor?
  - Steve's answer: You post to the jsdoc mailing list and ask if anyone else has had this problem. Several people say they have, and the issue pretty much dies right then and there.
  - Steve's real answer: Lexing and parsing

Rich Programmer Food, Steve Yegge

- Situation 3: You have to refactor a massive codebase in C++ in a non-trivial way, e.g. go from 32-bit to 64-bit. What do you do?
  - Steve's answer: You quit. Duh. You knew that was the answer before you reached the first comma.
  - Steve's real answer: Lexing and parsing

Rich Programmer Food, Steve Yegge

- Situation 4: You have to write a syntax highlighter for a web tool that deals with 5-8 programming languages.
  - Steve's answer: Tough it out. Colors are for weenies.
  - Steve's real answer: Lexing and parsing

Rich Programmer Food, Steve Yegge

- Situation 5: You have to communicate with a new router that has a telnet interface and a proprietary command language. You need to parse the responses in this language and look for patterns and produce new commands.
  - Steve's answer: Perl. It's a swiss army knife. You can use it to sidestep this problem with honor, by disemboweling yourself.
  - Steve's real answer: Lexing and parsing and codegen

Rich Programmer Food, Steve Yegge

- Situation 6: The “software engineers” at your company have decided to redesign the entire code base to make it easier to add to the codebase. How do you ensure things do not get worse?
  - Steve’s answer: Fire all the “software engineers”
  - Steve’s real answer: Lexing and parsing and codegen

Rich Programmer Food, Steve Yegge

- Situation 7: In order to remove a security hole in your Ruby on Rails website you have to make a set of non-trivial changes to the code to replace one idiom with another in your entire codebase.
  - Steve's answer: Fix it by hand. Hell, you only have about 10k lines of code for your whole site.
  - Steve's real answer: Lexing and parsing and codegen

Do you really know how programming languages work?

```
void
send (char *to, char *from, int count)
{
        while (count-- > 0)
                *to++ = *from++;
}
```

Different version of **send** with the same semantics

```
void
send2 (char *to, char *from, int count)
{
        int n = (count+7)/8;
        switch (count % 8) {
                case 0: while(n-- > 0) { *to++ = *from++;
                case 7:                  *to++ = *from++;
                case 6:                  *to++ = *from++;
                case 5:                  *to++ = *from++;
                case 4:                  *to++ = *from++;
                case 3:                  *to++ = *from++;
                case 2:                  *to++ = *from++;
                case 1:                  *to++ = *from++;
        }
    }
}
```

1. Is it valid C syntax? Will it compile?
2. What is it trying to accomplish?

Reason why **send2** works

- If you examine the C language specification for the language syntax specification,

  selection_statement -> SWITCH '(' expression ')' statement

  iteration_statement -> WHILE '(' expression ')' statement
  | DO statement WHILE '(' expression ')' ';'

  statement -> labeled_statement

  labeled_statement -> CASE constant_expression ':' statement

- **send2** is faster than **send** on some CPUs. Why?

Why should you take Compilers?

- If you want to be a good programmer
  - How does your code get converted into executable blobs
- If you want to design and implement "little" programming languages or "big" ones
  - Special purpose programming: for graphics, AI, etc.
- Because you want your CS education to matter
  - You don't want the hardest course you took to be Social Implications of CS, do you?
- Learn about software engineering that isn't boring

Answers to questions you always wondered about

- What kind of parser does the compiler for language X use?
  - Google it: Python uses LL(1) – what does that mean?
  - Why is LR better than LL – what in hell is that?
  - What is an abstract syntax tree?
- What is Apple's (or MS or Google) strategy for the next 5-10 years?
  - What is Clang?
- How does Google debug complex C++ code?
  - Answer: they modify the compiler, in fact they use Clang

Answers to questions you always wondered about

- What is at the core of modern web browsers?
  - http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/
  - You guessed it: browsers at their core do lexing & parsing
- Who should optimize code: programmer or compiler?
  - What kind of optimization can be done automatically?
- You thought formal languages were cool but useless
  - Beautiful theory leads to fast and clean code

Education ≠ Real Life?

Where: TASC1 9204
When: Tuesday, April 13, 2010 @ 1:00 PM
Who: Andrew Brownsword, Chief Architect from Electronic Arts BlackBox

Talk info:

Trajectories in Computing

Andrew looks at where computer hardware has been, how it has evolved to the present day, where it will go next, and what might happen after that. Along side the hardware trajectory he considers where programming has come from, where it has gone, and where it needs to go now and into the future. His perspective is solidly that of a practical and pragmatic industry software engineer, and his goal is to shamelessly interest everyone in how to help make his work easier in the future.

Quote from Andrew: "I wish I had taken Compilers during my undergrad degree."

“I talked about the compilers project at almost every interview I've had. “

-- Student who took CMPT 379 in Fall 2011
(now employed in the Bay Area)

Rich Programmer Food, Steve Yegge

"In fact, Compiler Construction is, in my own humble and probably embarrassingly wrong opinion, the second most important CS class you can take in an undergraduate computer science program."

Steve's most important CS class:
"Typing 101. Duh"