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)

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 …”

What do you learn?

1. Lexing – lexical analysis. Recognizing the tokens of the language.
2. Parsing – syntactic analysis, aka the structure of the program.
3. Type analysis – constraints on using the language.
4. Code generation and optimization.

Letters and words

Sentences

Semantics

Map to the real world

Rich Programmer Food (Steve Yegge)

- Situation 1: How do you auto-format source code of a huge Java library?
- Situation 2: Your company decides to do automatic documentation extraction from Javascript code. How do you write your own jsdoc extractor?

Rich Programmer Food (Steve Yegge)

- Situation 3: You must refactor a massive codebase in C++ in a non-trivial way, e.g. go from 32-bit to 64-bit. What do you do?
- Situation 4: You must write a syntax highlighter for a web tool that deals with 5-8 programming languages.
- Situation 5: You must communicate with a new router that has a telnet interface and a proprietary command language. You need to parse the responses and produce new commands.

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. You have to write them a tool to ensure code maintenance does not get worse?
- Situation 7: In order to remove a security hole you must make a set of non-trivial changes to the code to replace one idiom with another in your entire codebase. (Look up CVEs)

Do you really know how programming languages work?

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

```
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++;
                }
        }
}
```

Compare send and send2

- Q1: Is this valid C syntax?
- Q2: Why re-write send (2 lines of code) as send2 (10 lines of code). Is there a reasonable purpose behind send2?

Why should you take Compilers?

- Understand how programming languages work from the inside-out
- Design and build your own programming language (video games, AI, robotics, security, GPUs, concurrency)
- Contribute to development of an existing programming language e.g. faster javascript in a web browser

Why should you take Compilers?

- Write tools that can transform programs into other programs
- Understand parsing algorithms that take text input and transform it into tree structures
- Understand code generation and code optimization
- Be fluent in compiler tools like lex, yacc, LLVM

Extra Slides

Compare send and send2

- Q1: Is this valid C syntax?
  - Write down send2 with a main function and compile it. Does it work?
  - Examine the C language specification for switch and while loops. Does the code match the specification?

Compare send and send2

- Q1: Is this valid C syntax?
  - C language specification

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

iteration_statement -> WHILE '(' expression ')' statement

                     | DO statement WHILE '(' expression ')' ';'

statement -> labeled_statement

labeled_statement -> CASE constant_expression ':' statement
```

Compare send and send2

- Q2: Why re-write send (2 lines of code) as send2 (10 lines of code). Is there a reasonable purpose behind send2?
  - Compile and run two programs: one with send and one with send2
  - Are they executing the same instructions?
  - Are they executing the instructions the same number of times?
  - Is one faster than the other?

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"