CSE130: Programming Languages

Winter 2018
Tue&Thur 6:30-7:50 PM
Deian Stefan
Your instructor

• Assistant Professor in CSE

• Research: building secure systems
  ➤ Security + PL + Systems

• Industry: startup building secure runtime for Node.js
  ➤ Lots of PL ideas appear in daily work
Your TAs

• Abdulrahman Alkhelaifi

• Nadah Feteih

• Purag Moumdjian

• Kaiser Pister

• Sanjeev T Reddy
What is CSE 130 about?
What this course is not about?

• Learning how to write...
  ➤ JavaScript in January
  ➤ Haskell in February
  ➤ C++ in March
  ➤ etc.

• Learning C++, JavaScript, etc. to spec
What this course is about

• Concepts in programming languages
  ➤ Fundamentals and core features and building blocks
  ➤ Different programming paradigms and their use

• Design and implementation of languages
  ➤ Goals and trade-offs (with historical context)
  ➤ The cost of a language feature
What this course is about

• Concepts in programming languages
  ➤ Fundamentals and core features and building blocks
  ➤ Different programming paradigms and their use

• Design and implementation of languages
  ➤ Goals and trade-offs (with historical context)
  ➤ The cost of a language feature
Why?

• Concepts in programming languages
  ➤ Language shapes your thinking! Language features dictate how we express ideas and computation
  ➤ E.g., think of error handling in C vs. Java

• Design and implementation of languages
  ➤ Nothing is free: understand what you’re giving up and what you’re gaining when choosing a language
  ➤ E.g., exception handling, garbage collection, etc.
Why?

• Concepts in programming languages

➤ Language shapes your thinking! Language features dictate how we express ideas and computation

➤ E.g.,

This program prints “Hello World!”:

```

+--------+-..+++.>>.<-.<++.------.---><+.-++.
```

• Design and implementation of languages

➤ Nothing is free: understand what you’re giving up and what you’re gaining when choosing a language

➤ E.g., exception handling, garbage collection, etc.
Why?

• Concepts in programming languages
  ➤ Language shapes your thinking! Language features dictate how we express ideas and computation
  ➤ E.g., think of error handling in C vs. Java

• Design and implementation of languages
  ➤ Nothing is free: understand what you’re giving up and what you’re gaining when choosing a language
  ➤ E.g., exception handling, garbage collection, etc.

This program prints “Hello World!”:

```
++++++++[>++++[>+++++++>+++++++++++>>]>]>---.>
+++.+++.>-.<.+++.-------.-------.>++.>++.
```

https://en.wikipedia.org/wiki/Brainfuck
Why else?

- You can learn any of those languages... once you have a grasp of the fundamentals and understand features.

- You’ll usually want to use the right lang for the job... this ultimately comes down to what features you need.

- You will be able to think about programs differently... since you will understand what’s going on underneath.

- You will be in better shape to design and implement new languages... great features ➔ great language!
Why else?

• You can learn any of those languages... once you have a grasp of the fundamentals and understand features

• You’ll usually want to use the right lang for the job... this ultimately comes down to what features you need

• You will be able to think about programs differently... since you will understand what’s going on underneath

• You will be in better shape to design and implement new languages... great features ➔ great language!
Why else?

• You can learn any of those languages... once you have a grasp of the fundamentals and understand features

• You’ll usually want to use the right lang for the job... this ultimately comes down to what features you need

• You will be able to think about programs differently... since you will understand what’s going on underneath

• You will be in better shape to design and implement new languages... great features ➔ great language!
Why else?

• You can learn any of those languages... once you have a grasp of the fundamentals and understand features

• You’ll usually want to use the right lang for the job... this ultimately comes down to what features you need

• You will be able to think about programs differently... since you will understand what’s going on underneath

• You will be in better shape to design and implement new languages... great features ➡️ great language!
I’ll be working on languages?

• Lots of systems have their own languages or have a language runtime system at their core:
  ➤ Editors (Lisp for Emacs, JavaScript for Atom)
  ➤ DBs (SQL, MongoDB’s JavaScript, …)
  ➤ PDF viewers (JavaScript!?)

• PL is hot! Likely to work on something new in industry
  Flow, React @ Facebook    Rust, Emscripten @ Mozilla,
  TypeScript @ Microsoft    Swift @ Apple    CUDA @ NVIDIA
I’ll be working on languages?

• Lots of systems have their own languages or have a language runtime system at their core:
  ➤ Editors (Lisp for Emacs, JavaScript for Atom)
  ➤ DBs (SQL, MongoDB’s JavaScript, …)
  ➤ PDF viewers (JavaScript!?)

• PL is hot! Likely to work on something new in industry
  Flow, React @ Facebook    Rust, Emscripten @ Mozilla,
  TypeScript @ Microsoft    Swift @ Apple    CUDA @ NVIDIA
If nothing else...

You can put Haskell and Rust on your resume!
Syllabus: The great ideas [Ramsey]

**Expressive power (say more with less)**
- First-class functions
- Type inference
- Monads
- Pattern matching
- Exception handling
- Continuations

**Reliability and reuse**
- Type polymorphism
- Modules
- Type classes
- Objects & inheritance

**Cross-cutting concerns**
- Memory management
- Concurrency
The great ideas \textbf{[JavaScript]}

**Expressive power (say more with less)**

- First-class functions
- Type inference
- Monads
- Pattern matching
- Exception handling
- Continuations

**Reliability and reuse**

- Type polymorphism
- Type classes
- Modules
- Objects & inheritance

**Cross-cutting concerns**

- Memory management
- Concurrency
The great ideas [Haskell]

Expressive power (say more with less)

First-class functions
Type inference
Monads
Pattern matching
Exception handling
Continuations

Reliability and reuse

Type polymorphism
Modules
Type classes
Objects & inheritance

Cross-cutting concerns

Memory management
Concurrency
The great ideas [C++]

Expressive power (say more with less)

- First-class functions
- Type inference
- Monads
- Pattern matching
- Exception handling
- Continuations

Reliability and reuse

- Type polymorphism
- Modules
- Type classes
- Objects & inheritance

Cross-cutting concerns

- Memory management
- Concurrency
The great ideas [Rust]

Expressive power (say more with less)
- First-class functions
- Type inference
- Monads
- Pattern matching
- Exception handling
- Continuations

Reliability and reuse
- Type polymorphism
- Modules
- Type classes
- Objects & inheritance

Cross-cutting concerns
- Memory management
- Concurrency
<table>
<thead>
<tr>
<th>Week of</th>
<th>Tuesday</th>
<th>Thursday</th>
<th>Friday (section)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 09</td>
<td>Intro and JavaScript crash course</td>
<td>High-order functions</td>
<td></td>
</tr>
<tr>
<td>Jan 11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jan 16</td>
<td>Lambda calculus</td>
<td>Lambda calculus (cont)</td>
<td></td>
</tr>
<tr>
<td>Jan 18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jan 23</td>
<td>Scope, storage management, function</td>
<td>Haskell crash course</td>
<td>Haskell crash course (cont)</td>
</tr>
<tr>
<td>Jan 25</td>
<td>implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jan 30</td>
<td>Type polymorphism and type inference</td>
<td>Type polymorphism and type inference (cont)</td>
<td>Midterm review</td>
</tr>
<tr>
<td>Feb 01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 06</td>
<td><strong>Midterm</strong></td>
<td>Type classes</td>
<td></td>
</tr>
<tr>
<td>Feb 08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 13</td>
<td>Type classes (cont)</td>
<td>Objects</td>
<td></td>
</tr>
<tr>
<td>Feb 15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 20</td>
<td>Objects (cont)</td>
<td>vtables, subtyping, inheritance</td>
<td></td>
</tr>
<tr>
<td>Feb 22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 27</td>
<td>Control flow, continuations, monads</td>
<td>Control flow, continuations, monads (cont)</td>
<td></td>
</tr>
<tr>
<td>Mar 01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 06</td>
<td>Rust crash course</td>
<td>Concurrency (cont)</td>
<td>Rust crash course (cont)</td>
</tr>
<tr>
<td>Mar 08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar 13</td>
<td>Concurrency (cont)</td>
<td>Concurrency (cont)</td>
<td></td>
</tr>
<tr>
<td>Mar 15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Logistics & course mechanics
Contact information

• Course website: [http://cse130.programming.systems](http://cse130.programming.systems)
  ➤ Goto place for links and resources

• Piazza: [https://piazza.com/ucsd/winter2018/cse130](https://piazza.com/ucsd/winter2018/cse130)
  ➤ Use this for general discussions and questions

• Staff email: [ucsd-cse130-winter18@googlegroups.com](mailto:ucsd-cse130-winter18@googlegroups.com)
  ➤ Use this if you need to get in touch with us directly
Contact information

• Course website: http://cse130.programming.systems
  ➤ Goto place for links and resources

• Piazza: https://piazza.com/ucsd/winter2018/cse130
  ➤ Use this for general discussions and questions

• Staff email: ucsd-cse130-winter18@googlegroups.com
  ➤ Use this if you need to get in touch with us directly
Contact information

• Course website: http://cse130.programming.systems
  ➤ Goto place for links and resources

• Piazza: https://piazza.com/ucsd/winter2018/cse130
  ➤ Use this for general discussions and questions

• Staff email: ucsd-cse130-winter18@googlegroups.com
  ➤ Use this if you need to get in touch with us directly
Logistics: Lectures & Section

• Lectures: Mondays and Wednesdays
  ➤ We will assign reading before every class
  ➤ Come prepared, bring clickers: we will ask questions during lecture

• Section: Fridays
  ➤ Come to section with questions!
  ➤ Goal: go over course material and problems similar to those assigned for homework
Logistics: Lectures & Section

• Lectures: Mondays and Wednesdays
  ➤ We will assign reading before every class
  ➤ Come prepared, bring clickers: we will ask questions during lecture

• Section: Fridays
  ➤ Come to section with questions!
  ➤ Goal: go over course material and problems similar to those assigned for homework
Reading from:

- Optional course textbook
  - Concepts in Programming Languages by John Mitchell
  - Renting: cheaper option
  - We’ll be distributing new Chapters
- Papers & online resources
Logistics: Participation [5%]

• In class: answer questions, ask questions
• OH: ask questions, answer questions
• Online: ask questions, answer questions

➢ I’ve give an additional 5% to the students who really put in the effort to help on Piazza
Logistics: Assignments [35%]

• Homework: roughly every week
  ➤ Work in **groups of 3** (but try to do it on your own first!), submit using gradescope
  ➤ Will be released Wednesdays
  ➤ Early deadline: following Tuesday night
    ➤ You get 10% of your grade if you turn it in early!
  ➤ Hard deadline: following Friday night
Logistics: Assignments [35%]

- Programming labs: roughly one every 2 weeks
  - Submit solution by yourself using gradescope
  - Will be released Fridays
  - Hard deadline: 2 weeks from the release date on Friday night
Exams [60%]

• Midterm exam: Feb 06, in class [25%]
  ➤ Can screw up; we’ll compute your score as:
    
    \[
    \text{midterm} > 0 \ ? \ \max(\text{final}, \text{midterm}) : 0
    \]
  
  ➤ Will reflect assignments, pretty straight forward

• Final exam: March 20, location and time TBA [35%]
  
  ➤ Will test you in new setting, expect to learn!
Summary: grading breakdown

• Participation: 5%
• Assignments: 35%
• Exams: 60%
Collaboration policy

• Talk with each other, talk on Piazza, use resources
  ➤ Collaboration is a good thing! Just credit the person or resource in your submission

• That said: I expect you to turn in your own work
  ➤ Don’t discuss particularities of a solution with others
  ➤ Don’t ask for a solution on StackOverflow and the like
  ➤ See academic integrity statement
Collaboration policy

• Talk with each other, talk on Piazza, use resources
  ➤ Collaboration is a good thing! Just credit the person or resource in your submission

• That said: I expect you to turn in your own work
  ➤ Don’t discuss particularities of a solution with others
  ➤ Don’t ask for a solution on StackOverflow and the like
  ➤ See academic integrity statement
Academic integrity, conduct, etc.

• Goal: welcoming class where all can learn and feel included, safe, healthy

➤ I don’t want to run the class like a police state, but these two rules will be enforced: these matter even once you graduate!

➤ Eat, sleep, take care of your health

➤ Talk to me if you’re concerned
Feedback wanted!

➤ How’s the pace?

➤ Are there particular topics you want to spend more time on?

➤ What can I do to make your learning experience better?
Questions?