ECE 2400 / ENGRD 2140
Computer Systems Programming
Course Overview

Christopher Batten
School of Electrical and Computer Engineering
Cornell University

http://www.csl.cornell.edu/courses/ece2400
ECE 2400 / ENGRD 2140
Computer Systems Programming

What is Computer Systems Programming?

Activity: Comparing Algorithms

Trends in Computer Systems Programming

Course Logistics
The Computer Systems Stack

Gap too large to bridge in one step (but there are exceptions, e.g., a magnetic compass)
In its broadest definition, computer engineering is the development of the abstraction/implementation layers that allow us to execute information processing applications efficiently using available manufacturing technologies.

Computer Engineering is at the interface between hardware and software and considers the entire system.

Traditional Computer Science

Traditional Electrical Engineering
## Python for Application-Level Programming

### Computer Engineering
- Application
- Algorithm
- Programming Language
- Operating System
- Compiler
- Instruction Set Architecture
- Microarchitecture
- Register-Transfer Level
- Gate Level
- Circuits
- Devices
- Technology

### Application-Level Software
- High-level, user-facing software
- Enable productively developing applications that provide new functionality to users
- Enable productively collecting, analyzing, visualizing data
- Sometimes called a productivity-level language
## C/C++ for System-Level Programming

<table>
<thead>
<tr>
<th>Computer Engineering</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algorithm</td>
</tr>
<tr>
<td></td>
<td>Programming Language</td>
</tr>
<tr>
<td></td>
<td>Operating System</td>
</tr>
<tr>
<td></td>
<td>Compiler</td>
</tr>
<tr>
<td></td>
<td>Instruction Set Architecture</td>
</tr>
<tr>
<td></td>
<td>Microarchitecture</td>
</tr>
<tr>
<td></td>
<td>Register-Transfer Level</td>
</tr>
<tr>
<td></td>
<td>Gate Level</td>
</tr>
<tr>
<td></td>
<td>Circuits</td>
</tr>
<tr>
<td></td>
<td>Devices</td>
</tr>
<tr>
<td></td>
<td>Technology</td>
</tr>
</tbody>
</table>

- Connects application software to the low-level computer hardware
- Enables carefully managing performance and resource constraints
- Sometimes called an efficiency-level language

### Connects
- Application-level software
- System-level software

### Enables
- Carefully managing performance and resource constraints

### Sometimes called
- Efficiency-level language
Dynamically Interpreted vs. Statically Compiled

The standard Python interpreter is called CPython and it is written in C!
Computer Systems Programming is Diverse

- Application
- Algorithm
- Programming Language
- Operating System
- Compiler
- Instruction Set Architecture
- Microarchitecture
- Register-Transfer Level
- Gate Level
- Circuits
- Devices
- Technology

- Python, MATLAB
- Ruby, Javascript
- SQL, LINQ
- NumPy
- GUI frameworks

- Interpreters
- Compilers
- Databases
- Numerical libraries
- Operating systems
- Embedded control
Aside: C/C++ for Application-Level Software

- Application
- Algorithm
- Programming Language
- Operating System
- Compiler
- Instruction Set Architecture
- Microarchitecture
- Register-Transfer Level
- Gate Level
- Circuits
- Devices
- Technology

Application-Level & System-Level Software
## A Tale of Two Programming Languages

### Python Programming Language
- Introduced: 1991
- Most of the machine details are hidden from programmer
- Programmer gives up some control for improved productivity
- Easily supports multiple programming paradigms
- Extensive standard library is included
- Slow and memory inefficient

### C/C++ Programming Language
- Introduced: 1972(C), 1979(C++)
- Most of the machine details are exposed to the programmer
- Programmer is in complete control for improved efficiency
- Easily supports multiple programming paradigms
- More limited standard library is included
- Fast and memory efficient
Comparing the Performance of Python vs. C/C++

<table>
<thead>
<tr>
<th>Language</th>
<th>Program Time / Fastest Program Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>C++</td>
<td>3</td>
</tr>
<tr>
<td>Rust</td>
<td>5</td>
</tr>
<tr>
<td>Fortran</td>
<td>10</td>
</tr>
<tr>
<td>Java</td>
<td>30</td>
</tr>
<tr>
<td>Swift</td>
<td>50</td>
</tr>
<tr>
<td>Go</td>
<td>100</td>
</tr>
<tr>
<td>Pascal</td>
<td>300</td>
</tr>
<tr>
<td>Lisp</td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td></td>
</tr>
<tr>
<td>Perl</td>
<td></td>
</tr>
<tr>
<td>Ruby</td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td></td>
</tr>
</tbody>
</table>
Program = Data Structure + Algorithm

While this course covers C/C++ and system-level programming, this course also builds off of your prior programming experience to further develop your understanding of data-structures and algorithms.

- **Data Structure**: Way of efficiently organizing and storing data along with methods for accessing and manipulating this data.

- **Algorithm**: Clear set of steps to solve any problem instance in a particular class of problems.
ECE 2400 / ENGRD 2140
Computer Systems Programming

What is Computer Systems Programming?

Activity: Comparing Algorithms

Trends in Computer Systems Programming

Course Logistics
Activity: Comparing Algorithms

**Application:** Sort 16 numbers

**Activity Steps**

1. Half the class will use Algorithm A, half uses Algorithm B
2. When instructor starts timer, flip over worksheet
3. Sort 16 numbers using assigned algorithm
4. Lookup when completed and write time on worksheet
5. Raise hand
6. When everyone is finished, then analyze data

**Algorithm A**

repeat 16 times
   find smallest number not crossed off in input list
   copy smallest number to next open entry in output list
   cross smallest number off input list
Activity: Comparing Algorithms

Algorithm B

repeat 8 times, once for each pair in column 1
  copy smallest into next open entry in next column
  copy largest into next open entry in next column

repeat 4 times, once for group of 4 in column 2
  repeat 4 times
    compare top two numbers not crossed off in both groups
    copy smallest number to next open entry in next column
    cross smallest number off input list

... and so on ...
ECE 2400 / ENGRD 2140
Computer Systems Programming

What is Computer Systems Programming?

Activity: Comparing Algorithms

Trends in Computer Systems Programming

Course Logistics
Trend towards IoT and Cloud w/ Novel Hardware

Roughly every decade a new, smaller, lower priced computer class forms based on a new programming platform resulting in entire new industries.
Trend towards IoT and Cloud w/ Novel Hardware

Roughly every decade a new, smaller, lower priced computer class forms based on a new programming platform resulting in entire new industries.

Cloud Computing
- Often requires low-latency, high-throughput to meet overall application requirements
- Increasingly w/ specialized HW

Internet-of-Things
- Very limited resource constraints (e.g., energy, memory)
- Requires carefully managing these resources to meet overall application requirements
- Increasingly w/ specialized HW
Example Application: Image Recognition

Starfish

Dog
Machine Learning (ML): Training vs. Inference

Training

many images

Model

forward

"starfish"

labels

backward

error

Inference

few images

"dog"
Computer Systems Programming in ML

Google TPU
- Training is done using the TensorFlow C++ framework
- Training can take weeks
- Google TPU is custom chip
- High-level ML frameworks use C++ under the hood

Movidius Myriad 2
- Custom chip for ML on embedded IoT devices
- Carefully crafted C/C++ ML libraries for inference
- Embedded control also in C/C++
ECE 2400 / ENGRD 2140
Computer Systems Programming

What is Computer Systems Programming?

Activity: Comparing Algorithms

Trends in Computer Systems Programming

Course Logistics
ECE 2400 is also an ENGRD and thus satisfies the engineering distribution requirement

ECE 2400 can be an excellent way to generally incorporate programming into your non-ECE engineering curriculum
Course Objectives

▶ **describe** both basic and advanced algorithms and data structures and how to analyze these algorithms and data structures in terms of time and space complexity

▶ **apply** the C/C++ programming languages to implement algorithms and data structures using different programming paradigms

▶ **evaluate** algorithm and data structure alternatives and make a compelling theoretical and/or practical argument for one approach

▶ **create** non-trivial C/C++ programs (roughly 1,000 lines of code) and the associated testing strategy from an English language specification

▶ **write** concise yet comprehensive technical reports that describe a program implemented in C/C++, explain the testing strategy used to verify functionality, and evaluate the program to characterize its performance and memory usage
Course Structure

- **Part 1: C Programming and Basic Data Structures & Algorithms**
  - variables, expressions, functions, conditional statements, iteration statements, recursion, static types, pointers, arrays, dynamic allocation, lists, vectors, complexity analysis, abstract data types, sequences, stacks, queues, sets, maps, sorting algorithms

- **Part 2: C++ Programming and Advanced Data Structures & Algorithms**
  - transition from C to C++, namespaces, flexible function prototypes, references, exceptions, new/delete, object oriented programming, generic programming, functional programming, concurrent programming, dynamic vs. static polymorphism, binary search trees, hash tables, graphs

- **Part 3: Systems Programming in the UNIX Environment**
  - POSIX standard library including I/O and processes

- **Format**
  - lectures, quizzes, discussion section, online discussion, readings, programming assignments, two prelim exams, final exam
Programming Assignments

► **PA1–4: Fundamentals**
  ▶ PA1: Math functions
  ▶ PA2: List and Vector Data Structures
  ▶ PA3: Sorting Algorithms
  ▶ PA4: Polymorphic Data Structures and Algorithms

► **PA5: System Software**
  ▶ PA5: Handwriting Recognition System

► **Every programming assignment involves**
  ▶ C/C++ “agile” programming
  ▶ State-of-the-art tools for build systems, version control, continuous integration, code coverage
  ▶ Performance measurement
  ▶ Short technical report
Course Staff

- Prof. Batten  Instructor
- Tuan Ta  ECE PhD
- Hongyi Deng  ECE MEng
- Yixiao Zhang  ECE senior
- Nicole Kwok  ORIE+CS senior
- Ho-Jung Yang  ECE junior
- Fareeza Hasan  ECE sophomore w/ CS minor
Frequently Asked Questions

▷ I have not taken CS 1110 nor CS 1112, can I take this class?
  ▷ We assume some basic programming experience, discuss with instructor

▷ ECE Majors – How does ECE 2400 satisfy degree requirements?
  ▷ ECE 2400 can count as your second ENGRD course
  ▷ ECE 2400 can count as an outside-ECE technical elective
  ▷ ECE 2400 satisfies the ECE advanced programming requirement

▷ CS Majors – Can I use ECE 2400 in place of CS 2110?
  ▷ No

▷ Other Majors – How does ECE 2400 satisfy degree requirements?
  ▷ ECE 2400 can count as one of your two required ENGRD courses

▷ Should I take both ECE 2400 and CS 2110?
  ▷ Sure!
Take-Away Points

- Computer systems programming involves developing software to connect the low-level computer hardware to high-level, user-facing application software and usually requires careful consideration of performance and resource constraints.

- We are entering an exciting era where computer systems programming will play a critical role in enabling both cloud computing and the internet-of-things.