Introduction to Computer Systems

Syllabus

Web Page

http://pdinda.org/ics

See the web page for more information.

Class discussions are on Piazza

We will make only minimal use of Canvas (grade reports, perhaps homework handin)

Instructor

Peter A. Dinda
Tech L463
pdinda@northwestern.edu
Office hours: Thursdays, 9-12, or by appointment

Teaching assistants

Conor Hetland
Ford 2-211 (Office Hours Held in Wilkinson Lab)
ConorHetland2015@u.northwestern.edu
Office hours: Sundays, 1-5 (in Wilkinson Lab), or by appointment

Maciej Swiech
Ford 2-211 (Office Hours Held in Ford Commons)
MaciejSwiech2007@u.northwestern.edu
Office hours: Mondays, 3-7 (in Ford Commons), or by appointment

Daniel Stein
danielstein2018@u.northwestern.edu
Office hours: Wednesdays, 3-4:30 and Fridays, 1:30-4 (in Wilkinson Lab), or by appointment

Location and Time

Lecture: Tuesdays and Thursdays, 2-3:20pm, Tech Auditorium
Recitation: Mondays, 7pm, Tech M345

Prerequisites

Required	EECS 211 or equivalent
Required	Experience with C or C++
Required	Some experience with programming in a Unix environment (e.g., as in EECS 211)

EECS 213 is a **required core course** in the Computer Science curriculum in both McCormick and Weinberg. It is also a required course for CS minors in both schools. 213 can also be taken for credit within the Computer Engineering curriculum. 300-level systems courses generally have 213 as a prerequisite.

Textbook

Randal E. Bryant and David R. O'Hallaron, *Computer Systems: A Programmer's Perspective, **Third Edition***, Prentice Hall, 2015, (ISBN-13: 978-0134092669, ISBN-10**:** 013409266X) (Required - Textbook)

- Details on http://csapp.cs.cmu.edu
- **Make sure you have the third edition of the book**. This edition is the first to focus on the 64 bit operation of the machine, which we will make extensive use of in this course, unlike previous instances of EECS 213.

Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language, Second Edition*, Prentice Hall, 1988 (ISBN 0-131-10370-9) (Reference)

- This remains the definitive book on C by its creators

Richard Stevens and Stephen Rago, *Advanced Programming in the Unix Environment*, *Third Edition*, Addison-Wesley, 2013 (ISBN-10: 0321637739 | ISBN-13: 978-0321637734) (Reference)

- This describes how to think like a Unix systems programmer
- The older editions, even the first edition, are very good

Objectives, framework, philosophy, and caveats

This course has four purposes. First, you will learn about the hierarchy of abstractions and implementations that comprise a modern computer system. This will provide a conceptual framework that you can then flesh out with courses such as compilers, operating systems, networks, and others. The second purpose is to demystify the machine and the tools that we use to program it. This includes telling you the little details that students usually have to learn by osmosis. In combination, these two purposes will give you the background to understand many different computer systems. The third purpose is to bring you up to speed in doing systems programming in a low-level language in the Unix environment. The final purpose is to prepare you for upper-level courses in systems.

This is a learn-by-doing kind of class. You will write pieces of code, compile them, debug them, disassemble them, measure their performance, optimize them, etc.

The specific computer architecture we will focus on in this class is the 64 bit Intel/AMD x86 architecture, which is used in virtually all supercomputers, clouds, clusters, servers, desktops, and laptop/notebook computers today.[1] The specific operating system we will use is Linux, which is used in most supercomputer, cloud, cluster, and server environments, and is the operating system of Android smartphones and ChromeBooks. The specific programming toolchain we will use is GCC (and GDB), which is an extremely widely used core toolchain on pretty much all platforms, except Windows. The ideas and concepts embodied in this architecture, operating system, and programming toolchain are commonly found in others.

This course is ideally taken after 211 early in your academic career.

Lectures

It is important that you complete the reading assigned for each lecture before the lecture. To the greatest extent possible, I would like to spend the time answering questions from the reading.

Discussion and Getting Help

In addition to lecture and office hours, the TAs will also hold an optional recitation section once a week. We will use an online discussion group on Piazza as well. There are no discussions on Canvas.

Resources

You will have Linux accounts on the Wilkinson and Tlab machines, and it should be possible to do a lot of your work on them, or other 64 bit Linux machines. However, you will also have access to a considerably more powerful server machine that can support many users simultaneously, and we expect most students will use that. We will test your labs on that machine.

Labs

We will have four programming labs. Their goal is to make you apply the concepts you've learned and to gain familiarity with Unix tools that can help you apply them. Labs should be done in groups of two.

[1] The 64 bit x86 architecture is also called "x86_64" and just "x64". You will also have a chance to look at the Xeon Phi, an important derivative of the x64. We may also look briefly at the ARM architecture used in iPhones/iPads and many Android devices. If this doesn't make sense to you yet, don't worry about it.

Homework

This course includes four graded homework assignments. These are intended to interact with your reading

Exams

There will be a midterm exam and a final exam. The final exam will not be cumulative.

Grading

50 % Programming labs (12.5% per lab)
10 % Homeworks (2.5% per HW)
20 % Midterm (covers first half of the course)
20 % Final (covers second half of the course)

Peter ultimately assigns all grades. If you have a problem with a grade, you are welcome to bring it up with either Peter or the TAs, but only Peter is empowered to change grades.

Late Policy

For each calendar day after the due date for a homework or a lab, 10% is lost. After 1 day, the maximum score is 90%, after 2 days, 80%, etc, for a maximum of 10 days.

Cheating

Since cheaters are mostly hurting themselves, we do not have the time or energy to hunt them down. We much prefer that you act collegially and help each other to learn the material and to solve development problems than to have you live in fear of our wrath and not talk to each other. Nonetheless, if we detect blatant cheating, we will deal with the cheaters as per Northwestern guidelines.

Schedule

Lecture	Date	Topics	Readings	Homework/Labs
	3/29	***NO CLASS (University Follows Monday Schedule)***		
1	3/31 Th	Mechanics, Introduction, overview of abstractions	Chapter 1	Data lab out
4/4 is the last day for adding courses or changing sections				
2	4/5 T	Physics, transistors, photolithography, Moore's Law, bits, bytes, logic, cores, and multicores	2, 2.1, handout	HW1 out
3	4/7 Th	Integers and integer math	2.2-2.3	
4	4/12 T	Floating point	2.4-2.5	
5	4/14 Th	The Machine Model –	3, 3.1-3.5,	HW 1 in, HW 2

		instruction set architecture, microarchitecture, and basic instructions	5.7	out
6	4/19 T	Control flow	3.6	Data lab in Bomb lab out
7	4/21 Th	Procedures	3.7	
8	4/26 T	Data	3.8-3.10	
9	4/28 Th	Floating point	3.11-3.12	HW2 in, HW 3 out
Midterm Exam Review In Recitation				
10	5/3 T	Memory and cache	6, 6.1-6.4	
Midterm Exam: Wednesday, May 4, 6pm, Annenberg Hall G21, 1.5 hours, covering lectures 1-9, all related reading, etc.				
11	5/5 Th	Cache performance	6.5-6.7	Bomb lab in, Attack lab out
5/6 is the last day to drop a class				
12	5/10 T	Linking	Chapter 7	
13	5/12 Th	Exceptional control flow	8,8.1-8.4	
14	5/17 T	Exceptional control flow	8.5-8.8	HW 3 in
15	5/19 Th	Virtual memory Memory system	9, 9.1-9.8	Attack lab in, SETI lab out
16	5/24 T	Memory allocation	9.9-9.12	HW 4 out
17	5/26 Th	Input and Output	Chapter 10	
18	5/31 T	Network programming	Chapter 11 Handout	
19	6/2 Th	Concurrency, Distributed Systems and Wrap-up	Chapter 12 handouts	SETI lab in , HW 4 in
Finals week – Exam is Monday, June 6, 9 AM				

Note that in the latter part of the course, we will cover Chapters 10-12 at a very high level. I want you to read these chapters, but I will not cover them in their entirety in class.

We will skip Chapter 4 (Processor Architecture), 5 (Performance Optimization), and others. Chapter 4 is worth reading if you're interested in how a simple processor with an Intel-like instruction set is implemented. Chapter 5 is all about understanding how to make programs run faster.