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 enroll you.

We will make only minimal use of Canvas (grade reports, mostly)

Instructor

Peter A. Dinda
Mudd 3507
pdinda@northwestern.edu

Teaching Assistants

Conor Hetland
Mudd 3301
ConorHetland2015@u.northwestern.edu

Gino Wang
ginowang.sh@u.northwestern.edu

Nate Tracy-Amoroso
NathanielTracy-Amoroso2021@u.northwestern.edu

Office hours will be determined, with student input, during the first week.

Location and Time

Lectures:	Tuesdays and Thursdays, 2-3:20pm, Tech M345
Discussion:	Wednesdays, 4-4:50pm, Tech L361
Midterm Exam:	TBD, midquarter, outside of class
Final Exam:	Monday, 12/10, 12-2, 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.
- If you buy a non-U.S. version, acquire a pdf through some means, etc, please be aware that these can have differences from the U.S. version. In particular, for any homework question assigned from the book, please be sure to use a U.S. version. The U.S. version should be available in the library.
- There is now an electronic version of this book available for rent. For details, see the csapp web site.

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 / Attendance Requirement

It is important that you complete the reading assigned for each officially scheduled class session before that session (the reading for the first session is an exception). **Based on your reading, you should prepare at least one question for each class session.**

You are required to attend lecture. We use some materials and structure that are different from other instances of 213, and we do not use slides in lecture (you can find a pointer to the CMU slides on the web site).

In lieu of taking attendance, and to encourage you to do your reading, homework, and labs, as well as to broaden participation, **I will randomly choose students to call on**. If I call on you, I expect you to be there, and to ask a question, answer a question, or otherwise contribute to keeping the discussion going. That does not mean I expect the right answer, the right question, or the right comment – I just want a good faith effort that reflects your understanding of the reading and of the discussion so far.

What I'm asking of you is: Read. Attend. Ask. Answer. There is no such thing as a dumb question (or too esoteric of a question) - we will try our best to answer or comment on all questions.

Discussion / Attendance Requirement and Getting Help

Your TA and peer mentors will run a weekly discussion, which **you are required to attend.**

[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.

The structure of the discussion will depend on the week, and what we perceive student needs to be. In large part, though, we intend these to be a place for you to ask questions and get help. Conor, Gino, and Nate have all been in your shoes, having taken EECS 213 here at Northwestern, as well as more advanced courses that build on it. Conor has TAed 213 before.

Your instructor, TA, and peer mentors will also have regularly scheduled office hours and be available by appointment if these do not work. We will schedule office hours in the first week to maximize opportunities to attend.

We will use an online discussion group on Piazza as well. The intent is to have multiple venues for discussion with different styles so that all students feel comfortable participating. If you have a question, answer, or comment, please put it forward. If you're too scared, put it forward anonymously on Piazza.

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 considerably more powerful, and interesting server machines that can support many users simultaneously, and we expect most students will use those servers. We will test your labs on those machines.

For students who find the topics of this course particularly compelling, we can give you access to even more compelling machines.

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. **Start looking for a partner on day one.**

Homework

We will give you several homework assignments, along with solutions. These will not be graded, but we encourage you to do them since they are good for understanding the concepts in the book, and as preparation for the exams.

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)
20 % Midterm (covers first half of the course)
20 % Final (covers second half of the course)
10 % Participation (attendance, preparation, questions)

For some of the programming labs, extra credit is possible.

Your score in the course is the weighted average of your scores on each of the components. You can view all currently graded material, and your score, at any time on Canvas. Final grades are based on the course score (the weighted average), with the basic model being that the 90s are A territory, 80s are B territory, and so on. This model will be adapted toward lower thresholds if necessary based on overall class performance. That is, this is NOT a curved class.

The instructor ultimately assigns scores and grades in consultation with the TA and peer mentors. If you have a problem with a score on an assignment/exam or your grade, you are welcome to bring it up with him or the TA, but only the instructor is empowered to change grades.

Late Policy

For each calendar day after the due date for 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 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.

If you decide to place your work in a public repository (github, bitbucket, etc), be sure that you mark the permissions so that only you and your teammates can access it.

Please do not place class materials from on any public site. If it's on the course web site, it's already public and will remain public. If it's from the discussion group or from the handout directory on the course servers, it should not be shared publicly.

Schedule

Lecture	Date	Topics	Readings	Homework/Labs
1	9/27 Th	Mechanics, Introduction, overview of abstractions	Chapter 1	Data lab out
2	10/2 T	Physics, transistors, photolithography, Moore's Law, bits, bytes, logic, cores, and multicores	2, 2.1, handout	HW1 out
10/3 is the last day for adding courses or changing sections				
3	10/4 Th	Integers and integer math	2.2-2.3	
4	10/9 T	Floating point and FP math	2.4-2.5	
5	10/11 Th	The Machine Model – instruction set architecture, microarchitecture, and basic instructions	3, 3.1-3.5, 5.7	HW 2 out
6	10/16 T	Control flow	3.6	Data lab in Bomb lab out
7	10/18 Th	Procedures	3.7	
8	10/23 T	Data	3.8-3.10	
9	10/25 Th	Floating point *Instructor out of town; Guest lecture*	3.11-3.12	HW 3 out
10	10/30 T	Memory and cache	6, 6.1-6.4	
Midterm Exam Review: TBD, probably in discussion section				
Midterm Exam: Around here, time+location TBD				
11	11/1 Th	Cache performance	6.5-6.7	Bomb lab in, Attack lab out
12	11/6 T	Cache performance / catchup	6.5-6.7	
11/2 is the last day to drop a class				
13	11/8 Th	Linking	Chapter 7	
14	11/13 T	Concurrency and Parallelism	Chapter 12 (focus on 12.3+), handouts	
15	11/15 Th	Exceptional control flow	8,8.1-8.4	Attack lab in, SETI lab out
16	11/20 T	Exceptional control flow	8.5-8.8	HW 4 out
Thanksgiving break				
17	11/27 T	Virtual memory Memory system	9, 9.1-9.8	

18	11/29 Th	Memory allocation	9.9-9.12	
19	12/4 T	Input and Output	Chapter 10	
20	12/6 Th	Network programming	Chapter 11 Handout	SETI lab in
Finals week – Exam is Monday, December 10, 12-2				

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.