Introduction to Computer Systems

Syllabus

Web Page

http://pdinda.org/ics

We will not be using Blackboard for anything in this course.

Instructor

Peter A. Dinda
Tech L463
pdinda@northwestern.edu
Office hours: Thursdays, 2-5pm or by appointment

Teaching assistants

Maciej Swiech
Ford 2-221
dotpyfe@u.northwestern.edu
Office hours: Tuesdays 10-1 or by appointment

Kyle Hale
Ford 2-221
kh@u.northwestern.edu
Office hours: Wednesdays 10-1, or by appointment

Location and Time

Lecture:	Mondays and Wednesdays, 2-3:20pm, Tech LR4
Recitation:	Mondays, 6 PM, Tech L361 (starts second week)

Prerequisites

Required	CS 211 or equivalent
Required	Experience with C or C++

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 have 213 as a prerequisite.

Textbook

Randal E. Bryant and David R. O'Hallaron, *Computer Systems: A Programmer's Perspective*, ***Second Edition***, Prentice Hall, 2010, (ISBN-10: 0136108040 ISBN-13: 978-0136108047) (Required - Textbook)

- Details on http://csapp.cs.cmu.edu
- Make sure you have the second edition of the book.

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

- This remains the definitive book on C by its creators

Richard Stevens and Stephen Rago, *Advanced Programming in the Unix Environment, Second Edition*, Addison-Wesley, 2005 (ISBN-10: 0201433079 | ISBN-13: 978-0201433074 |) (Recommended)

- This describes how to think like a Unix systems programmer
- Note that the third edition will be released in June – if you decide to buy this for the class, don't buy new.

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 Intel/AMD x86 architecture, which is used in virtually all cloud, cluster, server, desktop, and laptop/notebook computers today.[1] The specific operating system we will use is Linux, which is used in most cloud and server environments, and is also the foundation of the Android platform. The specific programming toolchain we will

[1] For the most part, the lecture and programming assignments in this class look at x86 processors in 32 bit mode. We will try to touch on x86 in 64 bit mode as well. 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.

use is GCC (and GDB), which is the core toolchain on pretty much all platforms, except Windows.

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

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 also use a Google discussion group for the class. The group is a persistent one – you can see class discussions dating back about five years. There is a lot of useful material that students, TAs, and profs have posted over the years. For anything like schedules, announcements, clarifications, make sure you are looking at posts from the current quarter.

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.[2] 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

There will be 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.

Homework

Four problem sets will be assigned. Their goal is to help you improve your understanding of the material. Homework should be done alone.

Exams

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

Grading

10 %	Homeworks (2.5% per homework)
50 %	Programming labs (12.5% per lab)
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
Note that Monday classes are held on Tuesday during the first week, so we will meet for the first time on Tuesday, 4/2				
1	4/2 T	Mechanics, Introduction, overview of abstractions using web request-response	Chapter 1	Data lab out
2	4/3 W	Physics, transistors, photolithography, Moore's Law, bits, bytes, logic, cores, and multicores	2, 2.1, handout	HW 1 out,
Last day for late registration: Monday, 4/8				
3	4/8 M	Integers and integer math	2.2-2.3	
4	4/10 W	Floating point	2.4-2.5	
5	4/15 M	The Machine Model – instruction set architecture, microarchitecture, and basic instructions	3, 3.1-3.5, 5.7	HW 1 in, HW 2 out
6	4/17 W	Control flow	3.6	Data lab in Bomb lab out
7	4/22 M	Procedures	3.7	
8	4/24 W	Data	3.8-3.12	
9	4/29 M	64 bit x86 and perhaps some ARM	3.13-3.15, possible ARM materials	HW 2 in, HW 3 out
Midterm Exam: TBD, but around this time – will be in the evening				
10	5/1 W	Memory and cache	6, 6.1-6.4	

11	5/6 M	Cache performance	6.5-6.7	Bomb lab in, Buffer lab out
12	5/8 W	Linking	Chapter 7	
Last day for class drops: Monday, Friday, 5/10				
13	5/13 M	Exceptional control flow	8,8.1-8.4	
14	5/15 W	Exceptional control flow	8.5-8.8	HW 3 in
15	5/20 M	Virtual memory Memory system	9, 9.1-9.8	Buffer lab in, Fourth lab out,
16	5/22 W	Memory allocation	9.9-9.12	HW 4 out
17	5/27 M	*Memorial Day (No Class)*		
18	5/29 W	Input and Output	Chapter 10	
19	6/3 M	Network programming	Chapter 11 Handout	
20	6/4 W	Concurrency, Distributed Systems and Wrap-up	Chapter 12 handouts	Fourth lab in HW 4 in
Finals week – Exam is Wednesday, June 12, 3-5pm				

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), 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.