

# 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  
Tech L463  
[pdinda@northwestern.edu](mailto:pdinda@northwestern.edu)

### Teaching Assistants

Conor Hetland  
Ford 2-211  
[ConorHetland2015@u.northwestern.edu](mailto:ConorHetland2015@u.northwestern.edu)

Georgios Tziantzioulis  
Tech L466  
[GeorgiosTziantzioulis2011@u.northwestern.edu](mailto:GeorgiosTziantzioulis2011@u.northwestern.edu)

David Samuels  
[DavidSamuels2018@u.northwestern.edu](mailto:DavidSamuels2018@u.northwestern.edu)

William Wallace  
[WilliamWallace2018@u.northwestern.edu](mailto:WilliamWallace2018@u.northwestern.edu)

Murphy Angelo  
[MurphyAngelo2019@u.northwestern.edu](mailto:MurphyAngelo2019@u.northwestern.edu)

We will decide on office hours in the first week of class based on student feedback.

## Location and Time

Sessions: Tuesdays and Thursdays, 2-3:20pm, Tech L251  
Special Sessions: TBA

## 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.<sup>1</sup> 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.

### **Class Sessions / Question Requirement / 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). This is because, to the greatest extent possible, instead of flat out lecturing to you, I want to spend class time asking and answering questions about the readings, homework, and labs. This is a “flipped classroom” model where the readings replace the traditional lecture.

To better help student engagement with the reading materials drive class time, **you will need to contribute at least one question online by noon** of the day of the class session. Your question should derive from the reading materials of that day, as shown later in this document. You are also welcome to ask additional questions relating to reading, homework, labs, etc. I strongly encourage you to ask questions in other venues as well, including Piazza (which allows complete anonymity), and of course in person.

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

---

<sup>1</sup> 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.

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. Talk.

## Discussion and Getting Help

In addition to class sessions and office hours, the TAs will also hold occasional special topics sessions. Conor, George, David, Will, and Murphy have all been in your shoes, having taken EECS 213 here at Northwestern, as well as more advanced courses that build on it. Conor and George have both TAed 213 before.

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.

## 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)
- 10 % Questions (online and during class)
- 20 % Midterm (covers first half of the course)
- 20 % Final (covers second half of the course)

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

For the “Questions” portion of the grade: Contribute one clear question based on the assigned reading before each class session. Be there and make a good attempt when I randomly call on you. If you do these two things, you will be fine.

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 assign scores and grades in consultation with the TAs. 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 TAs, 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 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
1	3/28 T	Mechanics, Introduction, overview of abstractions	Chapter 1	Data lab out
2	3/30 Th	Physics, transistors, photolithography, Moore's	2, 2.1, handout	HW1 out

		Law, bits, bytes, logic, cores, and multicores		
<i>3/31 is the last day for adding courses or changing sections</i>				
3	4/4 T	Integers and integer math	2.2-2.3	
4	4/6 Th	Floating point and FP math	2.4-2.5	
5	4/11 T	The Machine Model – instruction set architecture, microarchitecture, and basic instructions	3, 3.1-3.5, 5.7	HW 2 out
6	4/13 Th	Control flow	3.6	Data lab in Bomb lab out
7	4/18 T	Procedures	3.7	
8	4/20 Th	Data	3.8-3.10	
9	4/25 T	Floating point	3.11-3.12	HW 3 out
10	4/27 Th	Memory and cache	6, 6.1-6.4	
<i>Midterm Exam Review: Wed, 4/26, 6pm</i>				
<i>Midterm Exam: Thurs, 4/27, 6pm.</i>				
11	5/2 T	Cache performance	6.5-6.7	Bomb lab in, Attack lab out
12	5/4 Th	Cache performance / catchup	6.5-6.7	
<i>5/5 is the last day to drop a class</i>				
13	5/9 T	Linking	Chapter 7	
14	5/11 Th	Concurrency and Parallelism	Chapter 12 (focus on 12.3+), handouts	
15	5/16 T	Exceptional control flow	8,8.1-8.4	Attack lab in, SETI lab out
16	5/18 Th	Exceptional control flow	8.5-8.8	HW 4 out
17	5/23 T	Virtual memory Memory system	9, 9.1-9.8	
18	5/25 Th	Memory allocation	9.9-9.12	
19	5/30 T	Input and Output	Chapter 10	
20	6/1 Th	Network programming	Chapter 11 Handout	SETI lab in
<i>Finals week – Exam is Friday, June 9, 9 AM</i>				

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.