Introduction to Computer Systems

Syllabus

Web Page and General Information

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 other than grade reports

Northwestern wants a range of statements in a syllabus. We can use the advanced computer science concept of a pointer instead. Here it is: https://www.registrar.northwestern.edu/faculty-staff/syllabi.html
A recursive traversal of that page will give you all the statements that apply to you. To summarize them: be nice, don't cheat, follow the COVID-19 guidelines, don't record people without permission, and don't be afraid to ask us and others for help.

Office hours and discussion/recitation times will be set during the first week based on student input and will be available on the course web page. Our goal is to have every student be able to attend at least one office hour per week, and for the maximum possible number of students be able to attend the optional discussion/recitation.

Class size and content ultimately depends on TA support.

Instructor

Peter A. Dinda
Mudd 3507
pdinda@northwestern.edu

Teaching Assistant and Peer Mentors

Matthew von Allmen (TA)
Mudd 3018
matthewvonallmen2026@u.northwestern.edu

Elena Fabian (PM)
elenafabian2024@u.northwestern.edu

Thomas Filipiuk (PM)
thomasfilipiuk2024@u.northwestern.edu

Alex Kang (PM)
alexkang2024@u.northwestern.edu

Mitchell Lai (PM)
mitchelllai2022@u.northwestern.edu

Liam Patterson (PM)
liampatterson2024@u.northwestern.edu

Santi Roches (PM)
santiroches2023@u.northwestern.edu

Molly Schneck (PM)
mollyschneck2023@u.northwestern.edu

Evan Waite (PM)
evanwaite2024@u.northwestern.edu

Location and Time

Lectures:	Tuesdays and Thursdays, 2-3:20pm, Annenberg G15
Optional discussion:	TBD, weekly
Midterm Exam:	TBD, mid-quarter, outside of class, in person
Final Exam:	Monday, 12/5, 12pm, in person

Prerequisites

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

CS 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.
- 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.

This version of CS 213 is intended to be particularly good preparation for CS 343, Operating Systems.

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. If you would also like to see slides, you can find a pointer to CMU slides on our course web site. These slides mirror the book closely.

In lieu of taking attendance, and to encourage you to do your reading, homework, and labs, as well as to broaden participation, **we will occasionally randomly choose students to call on**. If we call on you, we 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 we expect the right answer, the right question, or the right comment – we 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.

Optional Discussion Session and Other Ways of Getting Help

Your TA and PMs will run an optional weekly discussion, which we will schedule, with your input, during the first week. The goal of the optional weekly

[1] The 64 bit x86 architecture is also called "x86_64" and just "x64". We may also look briefly at the ARM architecture used in iPhones/iPads and many Android devices, and/or the up-and-coming RISC-V architecture. ARM is also the architecture used in "Apple Silicon" (M1, etc) machines. RISC-V is an open, public architecture. If this doesn't make sense to you yet, don't worry about it.

discussion is to provide a place to learn more and to get help in a more structured way than office hours.

Your instructor, TA, and PMs 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. We will enroll you. The link is on the course web page. 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, you can put it forward anonymously on Piazza. We will try our best to answer. We may switch from Piazza to something else, but the principle will be the same.

Resources

Machines at Northwestern will be available for remote login. You will have access to several 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. You should also be able to work on labs on your own machine provided it is running a reasonably recent Linux.[2]

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

Homeworks

We will give you several graded homework assignments, give you some time to work on them and hand-in the results, and then provide solutions. Homeworks are to be done individually, and are important for preparing for exams. The precise grading criteria will be given at later time.

[2] If you would like to do this, but your machine uses Windows or MacOS (Intel Only), you can install virtualization software, and then install Linux in a virtual machine. We typically use VMware for this (Workstation on a Windows box, Fusion Pro on a Mac), but there are other tools. Ubuntu is a reasonably good choice of Linux, although the CS department's servers run Red Hat.

Exams

There will be a midterm exam and a final exam. The final exam will not be cumulative. I do not provide practice exams. Instead, we will schedule midterm and final exam review sessions. Exams are not returned.

Grading

50 %	Programming labs (12.5% per lab)
10 %	Homeworks (4 assignments, 2.5% each)
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.

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 PMs. If you have a problem with a score on an assignment/exam or your grade, you are welcome to bring it up with them or the instructors, but only the instructors are 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 and Inadvertent Disclosures

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, say), 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. In general, if we give you sensitive information (like an exam), it will be cryptographically and steganographically associated with your name.

Accessibility / ANU

Any student requesting accommodations related to a disability or other condition is required to register with AccessibleNU (accessiblenu@northwestern.edu; 847-467-5530) and provide professors with an accommodation notification from AccessibleNU, preferably within the first two weeks of class. All information will remain confidential.

Schedule

Lecture	Date	Topics	Readings	Homework/Labs
1	9/20 T	Mechanics, Introduction, overview of abstractions	Chapter 1	Data lab out
2	9/22 Th	Physics, transistors, photolithography, Moore's Law, bits, bytes, logic, cores, and multicores	2, 2.1, physics-to-logic handout	HW1 out
9/26 is the last day for adding courses or changing sections				
3	9/27 T	Integers and integer math	2.2-2.3	
4	9/29 Th	Floating point and FP math	2.4-2.5	
5	10/4 T	The Machine Model – instruction set architecture, microarchitecture, and basic instructions	3, 3.1-3.5, 5.7	HW 1 in, HW 2 out
6	10/6 Th	Control flow	3.6	Data lab in Bomb lab out
7	10/11 T	Procedures	3.7	
8	10/13 Th	Data	3.8-3.10	
9	10/18 T	Floating point	3.11-3.12	HW 2 in HW 3 out
10	10/20 Th	Memory and cache	6, 6.1-6.4	
Midterm Exam Review: TBD, probably in discussion section				
Midterm Exam: Around here, time+location TBD				
11	10/25 T	Cache performance	6.5-6.7	Bomb lab in, Attack lab out
12	10/27 Th	Cache performance / catchup	6.5-6.7	
10/28 is the last day to drop a class				
13	11/1 T	Linking	Chapter 7	

14	11/3 Th	Concurrency and Parallelism	Chapter 12 (focus on 12.3+), Concurr-ency and Parallelism handouts	
15	11/8 T	Exceptional control flow	8,8.1-8.4	Attack lab in, SETI lab out
16	11/10 Th	Exceptional control flow	8.5-8.8 Unix Nutshell handout	HW 3 in, HW 4 out
17	11/15 T	Virtual memory Memory system	9, 9.1-9.8	
18	11/17 Th	Memory allocation	9.9-9.12	
19	11/22 T	Input and Output	Chapter 10	
Thanksgiving break				
20	11/29 T	Network programming or Slack	Chapter 11 Sockets handout	
21	12/1 Th	Slack or special topic (ARM, RISC-V, Floating Point, etc)		SETI lab in, HW4 in
Finals week – Exam is Monday, 12/5, 12pm				

Note that in the latter part of the course, we will cover Chapters 10-11 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.