CS343: Operating Systems

Syllabus - Fall 2024

Course Staff

Instructor

Branden Ghena branden@northwestern.edu

TAs

Connor Selna connorselna2021@u.northwestern.edu

PMs

Blake Hu

Emily Wei

Ethan Havemann

Jason Lu

Kevin Hayes

Max Glass

Natalie Hill

Robert Pritchard

Overview

This course introduces you to the basic, foundational concepts and principles of operating systems, many of which generalize to other areas of computer science and engineering. You will learn many of these concepts and principles by applying them in practice on a modern machine though labs that are designed to put you in the shoes of a systems-level developer. OS (and systems more broadly) is very much a learn-by-doing kind of area.

The following concepts and principles are included:

- **OS Structure:** kernel, device drivers, file systems, network stacks, schedulers, system calls, libraries, toolchains, language virtual machines, user interface/shell, applications, etc.
- **OS Models:** monolithic kernel, microkernel, virtual machine monitor/hypervisor, jail/zone/container, exokernel, unikernel, ...
- **OS Abstractions:** thread, name space, address space, process, IPC, virtual machine, container, file, directory stream, plus abstraction design within the kernel (devices, file systems, ...)
- **Concurrency Sources:** multiprocessors, devices, interrupts, threads, processes, horror stories, ...
- **Concurrency Challenges:** memory system coherence/consistency, race conditions, deadlock, livelock, horror stories, ...
- **Concurrency Control:** interrupt control, atomics, spinlocks, critical sections, blocking vs waiting, mutexes, semaphores, condition variables, monitors, barriers, lockfree/waitfree models, plus typical synchronization problems such as producer-consumer, reader-writer, and dining philosophers.
- **Scheduling and Resource Management:** theory, FIFO, SJF, SRPT, dynamic priority (e.g. Unix), lottery, fixed priority, preemptive vs non-preemptive, real-time vs non-real-time, horror stories, ...
- **Virtual Memory:** hardware-software co-design, paging, swapping, segmentation and (possibly) current alternatives.
- **Device Drivers:** interrupts, DMA vs PIO, MMIO vs PMIO, atomics, hardware memory barriers, software memory barriers.
- **Protection and Security:** kernel/user mode, mode/ring transitions, role of encryption, interaction with virtual memory, horror stories.
- **Memory management:** page allocation versus heap allocation, garbage collection, allocation in special contexts (e.g. interrupt context), page replacement, working set.
- **File systems:** issues/interfaces, data structures on block devices, examples (V7, FAT+, ext2+)
- **Principles:** policy versus mechanism, orthogonality, worse-is-better, lazy evaluation, caching, end-to-end argument, mythical man-month, no silver bullet, hardware/software co-design

Note that we will prioritize among these points so that if more time is needed to cover a high priority topic, there may be less or no coverage of a lower priority topic.

The hardware environment that we will focus on is Intel/AMD machines running in 64-bit mode ("x86-64" or "x64"), which is the commonplace platform for systems ranging from laptops to supercomputers today[1]. Your lab work will be done on Linux in the C programming language[2]. Two of your labs (on concurrency and scheduling) will be done in user-level Linux. The remaining labs will be in the context of the Nautilus kernel framework ("NK"), a research kernel developed at Northwestern and other institutions. The experience you gain in NK will generalize to the Linux kernel, for the most part[3].

Textbook

Modern Operating Systems, ***Fourth Edition****,*
Andrew S. Tanenbaum and Herbert Bos*,*
Pearson, 2014, (ISBN-13: 978-0133591620, ISBN-10**:** 013359162X)

We also considered several other books for this course, which may be useful as further references:

- Remzi Arpaci-Dusseau, and Andrea Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, 2018. [freely available]
- Abraham Silberschatz, Peter Galvin, and Greg Gagne, *Operating Systems Concepts,* 10th edition, Wiley, 2018.
- Thomas Anderson, and Michael Dahlin, *Operating Systems: Principles and Practice*, Recursive, 2014.
- William Stallings, *Operating Systems: Internals and Design Principles*, 9th edition, Pearson, 2017.

The CS213 textbook: *Computer Systems: A Programmer's Perspective* (Byrant & O'Hallaron) is also a good reference for this course as it provides a "what every programmer should know" treatment of some of the topics we will cover, including threads, processes, virtual memory, and the various Linux/Unix system call interfaces.

---

[1] Most of what you learn about x64 vis a vis OS will generalize to the other main platform, ARM, which is the basis for phones and tablets, as well as the next generation of Macs.

[2] Linux is the common OS on everything except laptops and desktops. It is also the OS underlying Android. C is the lingua franca of low-level software development.

[3] In the design of this course, we considered several other options. The most desirable would have been to have you work within the Linux kernel itself. This proved to be intractable from a pedagogical point of view. The complexity we would have to shield you from, particularly in a lab based on paging, would have been overwhelming to manage. We also considered the teaching OS xv6 for IA32 and for RISC-V. IA32 and RISC-V both would require revisiting material students have already learned, for x64, in CS213, plus xv6 for IA32 would have made a device driver lab particularly challenging to pull off. Another consideration was to use CMU's Pebbles OS specification and have students build Pebbles from scratch as in CMU's course. This was also limited to IA32, and seemed intractable to execute in a single quarter. The intent behind using NK is to give a view inside a modern, x64 codebase with clear internal interfaces that has a development model (e.g., Kbuild, C, etc) that is similar to Linux.

Location and Time

Lecture time: 12:30 - 1:50 PM Central, Tuesdays and Thursdays
Location: Fisk Hall, Room 217

I **strongly encourage** keeping up with the course lectures. Much of the difficulty in CS343 comes from the new concepts, which we take care to introduce in lecture. We will attempt to record all lecture sessions so that you can later review them if you want, but the expectation is that students will attend class.

Pre-requisites

CS213 or CE205 or equivalent: computer organization and introduction to systems
CS214 or equivalent: data structures
CS211 or equivalent: C programming experience and Unix environments

Any version of CS 213 or CE 205 is acceptable, but we will expect that you have seen basic concepts such as the existence of exceptional control flow and virtual memory, and the typical Unix system calls for processes, threads, and files+I/O. The syllabus shown in pdinda.org/ics is a good starting point.

Any version of CS 214 is acceptable, but we will expect that you have seen basic data structures, algorithms, and their implementation. These include linked lists, balanced search trees, hashing and hash tables, heaps, graphs, sorting, etc.

Experience with C or C++ in part means familiarity with arrays, structs, unions, and, most importantly, pointers and pointer-based data structures. Low-level pointer-based mechanisms are used throughout an OS, and by the underlying hardware.

CS343 satisfies one of the Systems Breadth, Tech Elective, and Project requirements in the Computer Science curriculum for both McCormick and Weinberg. CS343 also satisfies one of the Software Systems area requirements for the Computer Engineering curriculum in McCormick.

Communication

All course materials will be posted to Canvas including grades, lecture materials, and class recordings. Piazza will be used for course discussions and questions. **All questions should go to Piazza rather than to email.** We will enroll you in Piazza.

Office hours will also be available, with the regular schedule available on Canvas. Office hour appointments can also be made with the instructors or TA by Piazza post to “Individual Student(s) / Instructor(s) → Instructors”.

Class Structure

Schedule

The course schedule is available on the Canvas homepage for the course. Be aware that it is subject to change, although warnings will be given to students for any major changes.

Labs

We will have five programming labs. Except for the first lab, labs should be done in groups of up to three. **Start looking for partner(s) on day one.** We will provide a best-effort matching system for students who want partners but don't know anyone.

There are five labs. 60% of the grade in the class will be based on lab work, with a breakdown as follows:

5%	Getting Started Lab (done individually)
15%	Scheduling Lab
10%	Producer-Consumer Lab
15%	Device Driver Lab
15%	Paging Lab

We will use GitHub Classroom for disseminating and handing in labs. It is important that you and your partners make sure that your repositories are private. Only your group and the course staff should be able to see your repos.

The Producer-Consumer Lab and Queueing/Scheduling Lab are user-level Linux labs. The others are all done within a research kernel developed at Northwestern. All hardware is x86-64. All code is in C.

Midterm Exams

There will be two midterm exams. The exams will be given in person. One will take place near the middle of the class, with results returned to students before the drop deadline. The other will take place during the class exam slot of exam week.

Exams are taken individually. Students will be allowed the use of some amount of notes, which the instructor will provide guidance on in advance.

Midterm exams are not cumulative and focus on their own part of the course. However, the nature of the course is that the material in the second half builds on some knowledge from the first half.

Grades

Percentage grades will be converted to letter grades using the standard letter grade system (93% A, 90% A-, 87% B+, etc.). However, these grade bins may be moved at the instructor's discretion for the advantage of students. Note that the percent grade displayed by Canvas is not always accurate and may not take late penalties or slip days into account, as described below.

Each category of assignment has a total value, which is divided as explained above.

Category	Count	Total Value
Labs	5	60%
Midterm Exam 1	1	20%
Midterm Exam 2	1	20%

Late Policy

Midterm exams may not be taken late without prior coordination by the instructor.

Labs may be submitted late with a penalty of a 20% reduction in maximum points per day late. For example, a homework assignment submitted two days late has a maximum score of 60%. Lateness is rounded up to the whole day, so an assignment that is five minutes late has the same penalty as an assignment 23 hours late.

Slip Days

To help you more flexibly manage deadlines, we will give you **three slip days**, which allow you to submit a lab assignment late without penalty. Slip days are used in units of whole days, meaning a lab submitted five minutes late consumes an entire slip day. Slip days may only be applied to lab assignments, not exams.

You do not need to notify staff that you are using a slip day. We will track the total number of late days for your submissions and automatically apply slip days to optimize their usage. Slip days will not be assessed against lab assignments you did not submit. No extra credit is awarded for avoiding the use of slip days. However, it is in your best interest to avoid turning in lab assignments late, as the next assignment is often released slightly afterwards.

Slip days are applied individually, so for partner assignments be careful to communicate about plans to use slip days. It is possible for an assignment submitted one day late to have no penalty for one student (due to spending a slip day) and a one day late penalty for their partner with no slip days remaining.

Example slip day usage:

- Use two slip days to receive no penalty on a lab assignment submitted two days late.
- Use two slip days to receive no penalty on two separate lab assignments each submitted one day late.
- Use three slip days to receive just a one-day late penalty on a lab assignment submitted four days late.

Slip days are meant to automatically handle minor issues. If you are having a major issue, please contact the instructor as soon as possible and we will work together on a solution. Particularly for issues outside of the student's control, such as major injury, sickness, or family emergency, deadlines can be shifted without penalty if you contact the instructor.

Academic Integrity

Collaboration is a really good thing, and we encourage it. On the other hand, cheating is a very serious offense, which carries serious consequences. It is OK to meet with colleagues, form study groups, discuss assignments with them, compare alternative approaches, go over examples from textbooks or other sources. **But it is never ok to share code or homework solutions, or even to see each other's code or solutions.**

What you turn in must be your own work. Copying (or even studying) code, solution sets, etc., from anywhere (e.g., other people, web, GitHub, AI) is strictly prohibited. Be aware that we use a number of tools to detect and discover integrity violations. If you discuss your work with other people, please acknowledge them by listing their names in your submission. It is also forbidden to share, post, or otherwise publicize course materials. This includes (but is not limited to) homeworks, exams, solutions, or your own submissions. This extends even after the quarter ends; course material remains private information which you may not share or reproduce.

It is the responsibility of every student in this class to be familiar with and to adhere to the Academic Integrity Policies of Northwestern University and the McCormick School of Engineering. Any suspicion of violation of these policies will be reported immediately to the Associate Dean for Undergraduate Studies. If you are in doubt whether your actions constitute a violation of the above policies, ask the instructor (before doing what you are unsure about).

Sickness and Common Sense

Generally, if you are sick do not attend class. Instead contact your instructor as soon as possible and we'll figure out a way to handle the situation. I expect all students to use their discretion and make good choices for the community.

Accessibility

I believe in providing reasonable accommodations that allow for full access to learning for all. Please contact me for anything that might have an impact on your participation in this course (documented disability, language challenges, absences for religious observations, etc.).

Class Recordings

This class or portions of this class will be recorded by the instructor for educational purpose and available to the class during the quarter. Your instructor will communicate how you can access the recordings. Portions of the course that contain images, questions or commentary/discussion by students will be edited out of any recordings that are saved beyond the current term.

Diversity and Inclusion

I consider this classroom to be a place where you will be treated with respect, and I welcome individuals of all ages, backgrounds, beliefs, ethnicities, genders, gender identities, gender expressions, national origins, religious affiliations, sexual orientations, ability—and other visible and nonvisible differences. All members of this class are expected to contribute to a respectful, welcoming, and inclusive environment for every other member of the class.

Northwestern is committed to fostering an academic community respectful and welcoming of persons from all backgrounds. To that end, the policy on academic accommodations for religious holidays stipulates that students will not be penalized for class absences to observe religious holidays. If you will observe a religious holiday during a class meeting, scheduled exam, or assignment deadline, please let me know as soon as possible, preferably within the first two week of class. If exams or assignment deadlines on the syllabus fall on religious holidays you observe, please reach out so that we can discuss that coursework.

This course will also include a mix of undergraduates and graduate students with differing backgrounds in programming and computer systems. Do not feel discouraged by this. Each student will bring a different aspect of their knowledge to discussions, and we'll all be contributing towards increasing each other's understanding.

Northwestern University Syllabus Standards

This course follows the Northwestern University Syllabus Standards. Students are responsible for familiarizing themselves with this information.