COMPSCI 210 – Introduction to Computer Systems
Spring 2026

(Last Modified: January 9, 2026)

1 General

Course

Lecture	Wed/Fri 1:25 pm - 2:40 pm
Location	LSRC B101
Discussion	Mon (Multiple Times)

Instructors

Name	Matthew Lentz
Email	mlentz@cs.duke.edu
Website	https://users.cs.duke.edu/~mlentz/
Office Hours	(Briefly) After Lecture + TBD

Graduate TAs

Chunjiang Che chunjiang.che@duke.edu
Manmit Singh manmit.singh@duke.edu

Undergraduate TAs

Irene Biju
Thomas Kidane
Imani Kwesi
Joseph Loeffler
Amogh Manral
Anric Ngan
Sahil Patel
Islam Tayeb
Kevin Yao
Zaer Zaqy
Anna Zhang

Resources

Canvas	https://canvas.duke.edu/courses/71583
Gradescope	https://www.gradescope.com/courses/1204799
Ed	https://edstem.org/us/courses/90133
GitLab	https://coursework.cs.duke.edu/cs210d-s26

2 Overview

CPS 210 is an introduction to systems software and computer architecture, with programming exercises in the C language on the Linux operating system. This course provides a programmer's view of how computer systems execute programs and store information. It examines key computational abstractions behind high-level programming languages: number and data representations, memory/pointers, the stack, buffer overflow, processor instructions, caching, virtual memory, programs and processes, and basics of concurrency with threads. The Big Ideas in this class apply across subfields of computer systems and computer science.

3 Expectations

3.1 Preconditions

The prerequisite for this course is that you have completed CompSci 201 (or equivalent). Therefore, we expect that you have experience programming in a high-level language (e.g., Java)

3.2 Postconditions

After completing this course, we expect you to be able to:

1. **(Overall) Understand how software runs on real machines and operating systems, including implementation of selected programming language constructs (e.g., functions/procedures, loops, conditions)**
2. Understand the key programming language concepts within the C language (e.g., types, operators, control flow) and the surrounding ecosystem (e.g., compiler, linker, debugger)
3. Understand memory safety and memory protection, to establish concepts underlying computer security and modern safe programming languages
4. Understand the instruction set architecture (ISA) abstraction for CPU hardware and how C programs are mapped to machine-level instructions
5. Understand the role of the operating system (OS), OS abstractions provided to user programs (e.g., processes), and how underlying hardware enables them
6. Implement non-trivial programs in C that interact with details of the underlying machine architecture (e.g., stack layouts) and interfaces provided by the operating system (e.g., threads, mutexes)
7. Understand various factors that impact performance and resource efficiency as a result of hardware and software running underneath your programs (e.g., I/O, caching)

4 Resources

Textbooks

1. **[Required] Dive Into Systems**
   https://diveintosystems.org/book/index.html
2. **[Required] Operating Systems in Three Easy Pieces (OSTEP)**
   https://pages.cs.wisc.edu/~remzi/OSTEP/
3. **[Recommended] The C Programming Language**
   Kernighan and Ritchie
   2nd Edition

Canvas We will be using Canvas as a general course platform, primarily used for reporting grades.

Gradescope We will be using Gradescope to manage the submission and automatic grading for labs. We will also use it for grading exams.

Ed We will be using Ed to serve as a discussion forum for the course and the primary place for making course announcements.

Assistance We want to help you and we encourage you to visit office hours and/or ask questions on Ed as needed. Please be respectful of staff time: start early, seek help early, and do not expect extra time beyond the posted schedule. We expect you to make a good faith effort to solve problems yourself before seeking help; additionally, when asking for help, please explain your thought process and what you have tried so far so that we can better help you. Keep up with the Ed board and check to see if your question is already answered. If a TA refuses to help you, please respect their decision and raise the issue with the instructors if you feel you are treated unfairly.

We list office hours here, but for up-to-date information (e.g., cancellations, rescheduling) please refer to Ed.

5 Grading

Your final grade in the course will be determined by the following percentage allocations:

Type	%	Description
Exams	50	Midterm 1 (15), Midterm 2 (15), and Final (20)
Projects	20	Four programming assignments that span across labs/lectures
Labs	23	Small programming assignments associated with each lecture
Discussion	7	Participation and exercises

Note that lecture attendance is not mandatory; however, we strongly encourage it because there will be a number of interactive elements throughout the course. You are responsible for all material covered and assignments given out during any class that you miss.

5.1 Exams

There will be two midterm exams and one final exam. The midterm exams will be given in-class on February 18th and April 1st. The final exam is scheduled for Tuesday April 28th from 2-5pm in LSRC B101. Each exam is cumulative, covering all content from the start of the course up to the exam; however, for the midterm exams, content covered by the class immediately prior to each exam is out of scope (which is typically used for topic review).

For each exam, we allow you to make use of sheets of notes, with an incrementing number of sheets per exam (so you can make use of old ones as well). This means you can have one sheet for Midterm 1, two sheets for Midterm 2, and three sheets for the Final. These sheets may be double sided and may be hand-written or typed; however, all sheets must be on standard letter sized paper. With the exception of the note sheets above, the exam is closed book and closed notes.

5.2 Projects

There are four programming projects spread throughout the duration of this course. Each project builds on concepts and skills obtained from lectures, readings, and labs. You may work on projects individually or with at most one other person. Be sure to carefully read the README file associated with each project and follow directions appropriately.

5.3 Labs

There are many short (1-3 hours), individual programming labs associated with each lecture that help: 1) reinforce concepts learned from readings and lectures, and 2) prepare you for the projects. Labs are released after each lecture (Wednesday and Friday) and are typically due the following week. The Monday discussion sections will focus on the labs due that week; you should start working on both labs prior to your discussion section. Be sure to carefully read the README file associated with each lab and follow directions appropriately.

We will drop your three lowest lab scores.

5.4 Discussion

About half the discussion days on the schedule feature named group exercises (e.g., Performance). Attendance is required for these sessions to get credit.

Other discussion days are designated as *Labs (Optional)*, which are for assistance with labs that week. In contrast to previous offerings, there is no required attendance if you have not received full points prior to your discussion section starting. That said, it is your responsibility to make good use of these discussion sections. The TAs are there to help!

We allow one missed group exercise.

5.5 Late Policy

We expect you to turn in your work by the day and time it is due. Note that if a time is not listed, you should assume the deadline is 11:59pm ET on the day listed. However, for labs and projects, we allow you to turn in assignments up to 72 hours late with no penalty. This prevents the need to file any STINFs for assignments.

Exceptions to this late policy are allowed only for Dean's Excuses, which you can find more information about here: https://trinity.duke.edu/undergraduate/academic-policies/class-attendance-and-missed-work.

5.6 Regrading Policy

All regrading requests must be submitted within one week of the graded item being returned/available. Please use the regrade request functionality available on Gradescope to issue your request. Requests made later than one week will be denied.

6 Academic Integrity

We expect everyone to uphold the Duke Community Standard, which you can find here: https://studentaffairs.duke.edu/conduct/about-us/duke-community-standard. In particular, this standard is comprised of:

- I will not lie, cheat, or steal in my academic endeavors
- I will conduct myself honorably in all my endeavors
- I will act if the Standard is compromised

Please ask me if you are unsure which actions may (or may not) violate the community standard as part of this course. However, you can find specific collaboration guidelines for different types of coursework below.

6.1 Collaboration Guidelines

Labs must be completed individually. Projects may be completed individually or with a partner.

You may, and are very much encouraged to, discuss the projects and labs with your fellow classmates. This includes discussing the specification, proposed approaches to solving the assignment, working through high-level designs on a whiteboard, or asking questions on the forums. However, each of you is responsible for writing up your own implementations. Do *not* share code, either snippets or solutions as a whole, with one another; this also extends to the forums, unless there is explicit approval from the instructor or one of the TAs. Do not incorporate code you find on the Internet (or from paper archives), from students who took the class previous semesters, or other sources. The only exception to the above is when working on a project with a partner, in which case you are submitting a single implementation that you jointly authored.

You should add a text file (statement.txt) listing the nature and sources of any assistance that you received, as well as anything else you want to tell us. If you choose to work in a team of two for the project, you must add a reflection of the collaboration within your team to the file.

You can use LLMs (e.g., ChatGPT) and other AI-based tools to aid in understanding course material, but you must *not* use these tools to aid in writing source code. This includes using Github Copilot or similar

tools for suggestions as you code; these tools should be disabled (if installed) when completing work for this course.

The line should be quite clear here, but please reach out if you have any questions regarding this policy.

7 Students with Disabilities

Duke University is committed to providing equal access to students with documented disabilities. Students with disabilities may contact the Student Disability Access Office (SDAO) to ensure your access to this course and to the program. There you can engage in a confidential conversation about the process for requesting reasonable accommodations both in the classroom and in clinical settings. Students are encouraged to register with the SDAO as soon as they begin the program. Please note that accommodations are not provided retroactively. More information can be found online at access.duke.edu or by contacting SDAO at 919-668-1267, SDAO@duke.edu.

8 Environment

Interactive discussions are one of the key components that make this type of course useful, especially when we dive into more advanced topics. I want everyone to make sure that they do their best to foster an inclusive environment, since that will enable us to have the richest discussions. In general, please treat teaching staff and other students with kindness and respect both in class and outside of class (e.g., Ed forums). We will disable anonymous posting if we see any threatening or distruptive posts. If you feel uncomfortable for any reason, please let me know.

Let us know if you have concerns we can address regarding your safety or health. Please know that we are concerned for you. We understand that you may be facing negative reactions to stress and pressure, other personal challenges, or just the burdens of managing your life and future. Be mindful of your needs for sleep, exercise, proper food, recreation, social connection, and constructive engagement with your problems. We encourage you to take advantage of Duke resources for wellness and mental health. Ask for help when you need it.

9 Course Evaluations

Please take a moment of your time at the end of the semester to submit a course evaluation. These evaluations are incredibly useful to both us personally as well as to the department as a whole. You can provide your feedback at the following link: http://duke.evaluationkit.com/. Note that if you have suggestions for how we improve the course, feel free to reach out at any time.

10 Modifications

We tried to make this syllabus both correct and complete; however, we reserve the right to modify the contents of the syllabus while the course is underway. We will make sure that any modifications are clearly communicated to you with sufficient advance notice.