

# Preliminaries



CS 450: Operating Systems  
Michael Lee <lee@iit.edu>

# Instructor

- Michael Lee
  - Email: [lee@iit.edu](mailto:lee@iit.edu)
  - Office: SB 226C
  - Hours: Wed & Fri 1-3PM
    - By appointment only! (Zoom or In-person)

# Teaching Assistants

- Lan Nguyen <lnguyen18@hawk.iit.edu>
- Aashesh Kumar <akumar88@hawk.iit.edu>
- Office hours will be posted on course website

# Agenda

- Prerequisites
- Resources: website, textbooks, etc.
- Evaluation: assignments, exams, grading
- Class overview

# § Prerequisites

# CS Essentials

- Essential algorithms & runtime analysis
- Data structures
- Data representation (bin/hex) and manipulation (shifting/masking/etc.)

# Programming Knowledge

- Languages: Assembly (x86 or other), C (or other procedural)
- Compilation process (assembly, compilation, linking, etc.)
- Runtime stack usage and conventions
- Dynamic memory allocation

# Computer Organization

- Von Neumann model
- Instruction Set Architectures (RISC/CISC)
- Cache organization and operation
- Interrupt procedures



# Operating System API

- Ideally: knowledge of Unix syscalls
  - process management (fork/exec/wait)
  - memory management (sbrk/mmap)
  - I/O (open/close/read/write/seek)

# Support Tools

- Command line / Shell (e.g., bash)
- Debugger/Tracer (e.g., GDB)
- Build automation (e.g., Make)
- Version control (e.g., Git)

# § Resources

**CS 450: Operating Systems**

### Announcements

- Welcome to the Spring 2023 offering of CS 450: Operating Systems!

### Calendar

Please note that readings for a given lecture should ideally be reviewed *before* coming to class, and will likely need to be revisited afterwards. Most readings are from "[Operating Systems: Three Easy Pieces](#)" (OS:TEP). All materials can be found online, and are linked below. Lecture slides/notes will be posted after class, when available.

The lecture calendar is tentative and may be updated.

Date	Topic	Notes	Reading(s)
Jan 11	Syllabus and Course overview	<a href="#">01-prelim</a>	<a href="#">Syllabus</a>
Jan 13	Operating systems overview	<a href="#">02-what-is-an-os</a>	OS:TEP Chapters <a href="#">1</a> , <a href="#">2</a>
Jan 18	Processes	<a href="#">03-the-process</a>	OS:TEP Chapters <a href="#">3</a> , <a href="#">4</a> , <a href="#">5</a>
Jan 20	CPU virtualization	<a href="#">04-cpu-virt</a>	OS:TEP Chapter <a href="#">6</a>
Jan 25	xv6 & x86	<a href="#">05-x86-and-xv6</a>	<a href="#">xv6 Commentary</a> Chapters 0, 1, Appendices A, B
Jan 27	xv6 code review and demo		<a href="#">xv6 Source</a>
Feb 1 - Feb 8	Scheduling	<a href="#">06-scheduling</a>	OS:TEP Chapter <a href="#">7</a> and <a href="#">8</a>
Feb 10 - Feb 17	Queueing theory	<a href="#">07-queueing</a>	<a href="#">Queueing Systems</a> 1, 2, 3 <a href="#">Practice problems, solutions</a>
Feb 22	Virtual memory	<a href="#">08-vm-a</a>	OS:TEP Chapters <a href="#">12</a> , <a href="#">13</a> , <a href="#">14</a>
Feb 24	Segmentation and Paging	<a href="#">08-vm-b</a>	OS:TEP Chapters <a href="#">15</a> , <a href="#">16</a> , <a href="#">18</a> , <a href="#">19</a> .

Course website: <http://moss.cs.iit.edu/cs450>

Operating Systems: Three Easy Pieces  
 Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau  
 Arpaci-Dusseau Books  
 August, 2018 (Version 1.00)

And now, the free online form of the book, in chapter-by-chapter form (now with chapter numbers!):

Intro	Virtualization		Concurrency	Persistence	Security
Preface	3 <i>Dialogue</i>	12 <i>Dialogue</i>	25 <i>Dialogue</i>	35 <i>Dialogue</i>	52 <i>Dialogue</i>
TOC	4 Processes	13 Address Spaces <small>code</small>	26 Concurrency and Threads <small>code</small>	36 I/O Devices	53 <i>Intro Security</i>
1 <i>Dialogue</i>	5 Process API <small>code</small>	14 Memory API	27 Thread API <small>code</small>	37 Hard Disk Drives	54 <i>Authentication</i>
2 <i>Introduction</i> <small>code</small>	6 Direct Execution	15 Address Translation	28 Locks <small>code</small>	38 Redundant Disk Arrays (RAID)	55 <i>Access Control</i>
	7 CPU Scheduling	16 Segmentation	29 Locked Data Structures	39 Files and Directories	56 <i>Cryptography</i>
	8 Multi-level Feedback	17 Free Space Management	30 Condition Variables <small>code</small>	40 File System Implementation	57 <i>Distributed</i>
	9 Lottery Scheduling <small>code</small>	18 Introduction to Paging	31 Semaphores <small>code</small>	41 Fast File System (FFS)	
	10 Multi-CPU Scheduling	19 Translation Lookaside Buffers	32 Concurrency Bugs	42 FSCK and Journaling	<b>Appendices</b>
	11 <i>Summary</i>	20 Advanced Page Tables	33 Event-based Concurrency	43 Log-structured File System (LFS)	<i>Dialogue</i>
		21 Swapping: Mechanisms	34 <i>Summary</i>	44 Flash-based SSDs	<i>Virtual Machines</i>
		22 Swapping: Policies		45 Data Integrity and Protection	<i>Dialogue</i>
		23 Complete VM Systems		46 <i>Summary</i>	<b>Monitors</b>
		24 <i>Summary</i>		47 <i>Dialogue</i>	<i>Dialogue</i>
				48 Distributed Systems	Lab Tutorial
				49 Network File System (NFS)	Systems Labs
				50 Andrew File System (AFS)	xv6 Labs
				51 <i>Summary</i>	

**INSTRUCTORS:** If you are using these free chapters, **please just link to them directly** (instead of making a copy locally); we make little improvements frequently and thus would like to provide the latest to whomever is using it. Also: we have made our own class-preparation notes available to those of you teaching from this book; please drop us a line at [remzi@cs.wisc.edu](mailto:remzi@cs.wisc.edu) if you are interested.

**HOMEWORKS:** Some of the chapters have homeworks at the end, which require simulators and other code. More details on that, including how to find said code, can be found here: [HOMEWORK](#)

**PROJECTS:** While the book should provide a good conceptual guide to key aspects of modern operating systems, no education is complete without projects. We are in the process of making the projects we use at the University of Wisconsin-Madison widely available; an initial link to project descriptions is available here: [PROJECTS](#). Coming soon: the automated testing framework that we use to grade projects.

**BOOKS NEWS:** Lots of new stuff to finally get to version 1.0. Track changes: [NEWS](#)

**ACKNOWLEDGEMENTS:** These students have greatly contributed to this effort, through endless bug reports and other comments. Your name could go here! (as well as in the printed book): [ERRATA](#)

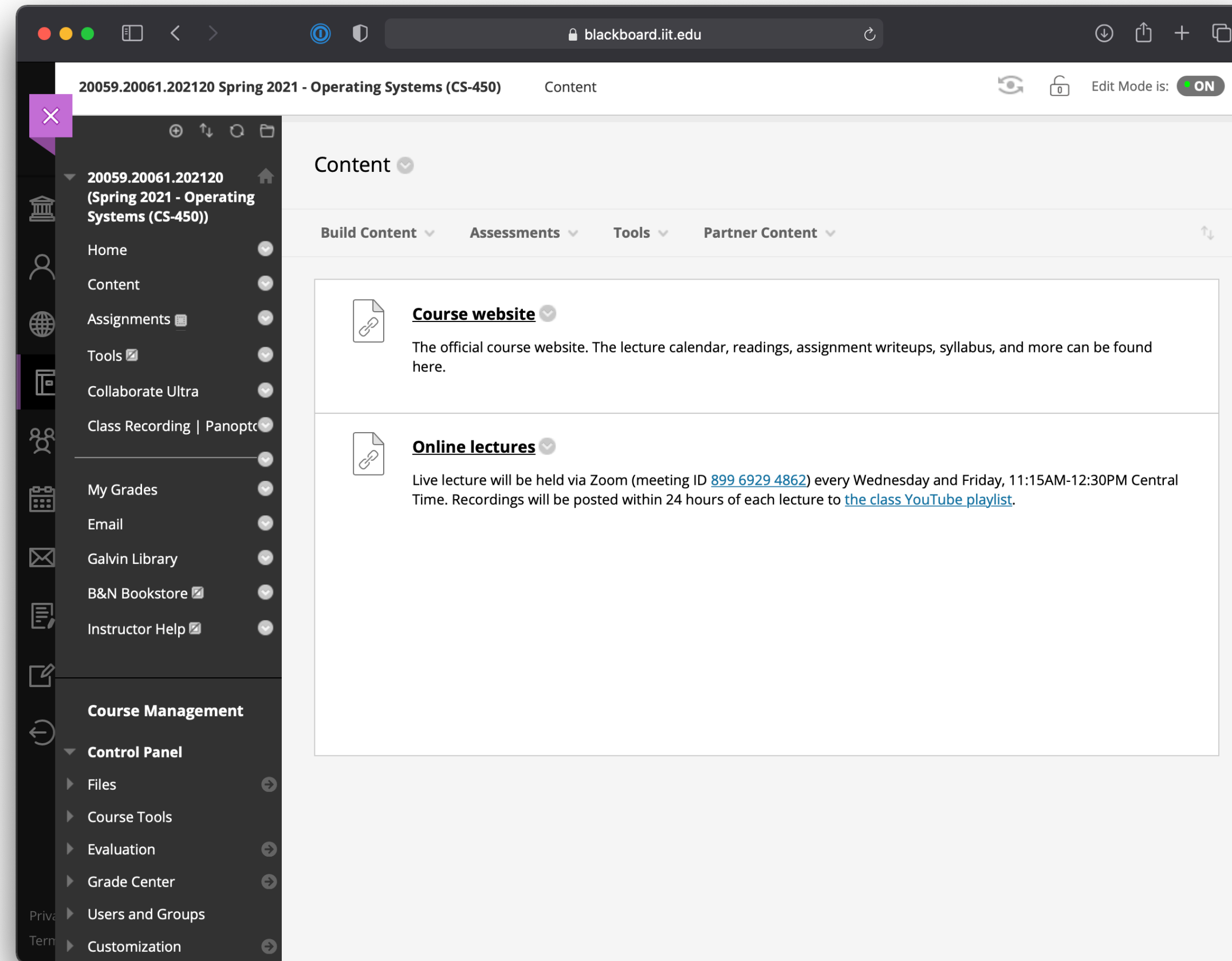
**OTHER SYSTEMS BOOKS:** Interested in other systems books? Good! Of course, we assume some background in [The C Programming Language](#), so that's a good investment. And [Advanced Programming in the UNIX Environment](#) is a must for any shelf. On top of that, here are some OS books that could be worth your time: [Operating](#)

# Textbook: “Operating Systems: Three Easy Pieces”

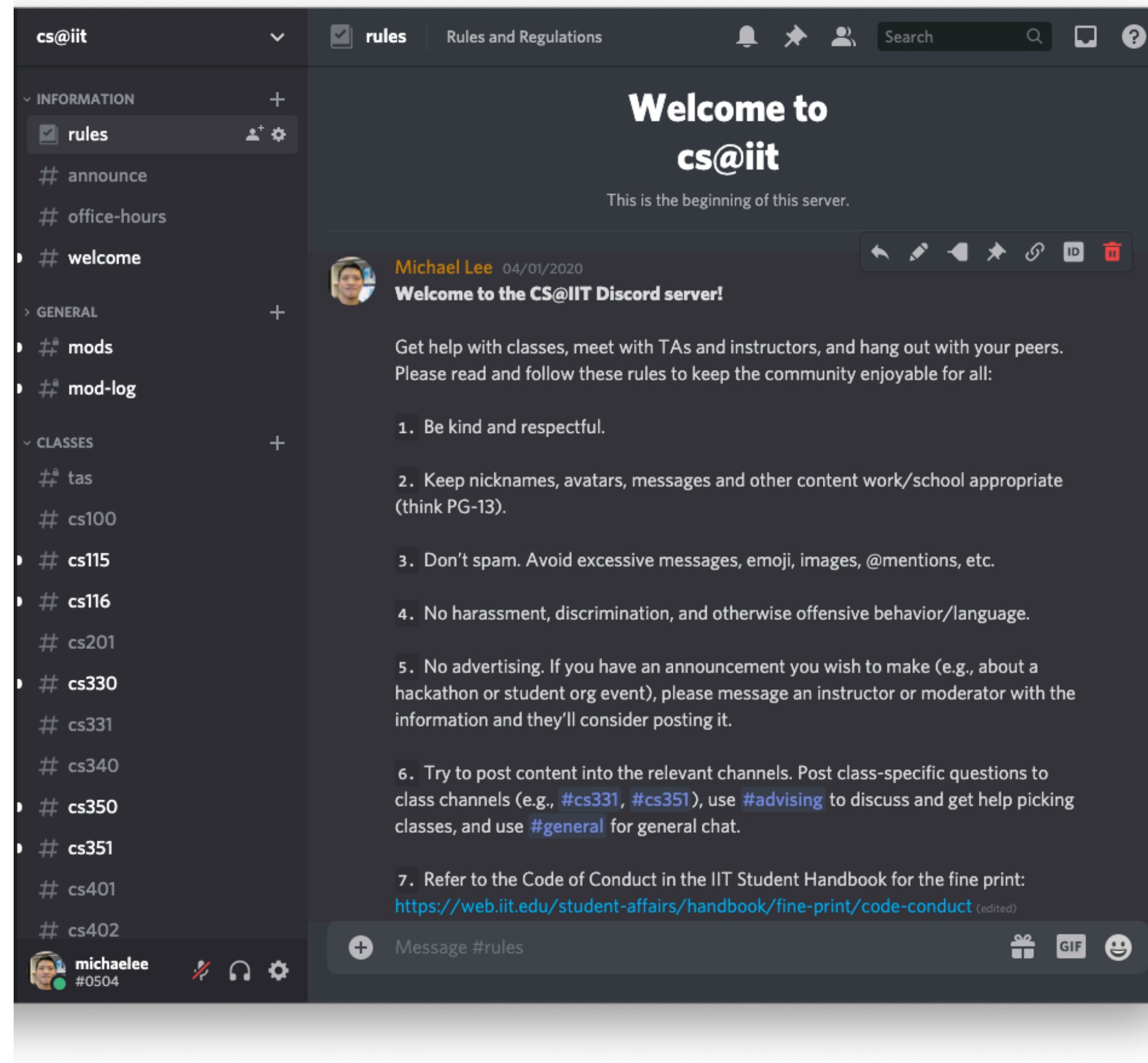
<http://pages.cs.wisc.edu/~remzi/OSTEP/>

# Readings before class!

- Please read (or at least, skim) readings *before* associated lecture
  - OS:TEP is *very* readable, especially for an OS text!
- Check website frequently in case schedule changes



Blackboard: <http://blackboard.iit.edu>



## Discord: Class discussion and Q/A



# § Grading / Evaluation

# Assignments

- 5-7 assignments = 60% of grade:
  - Problem sets (quantitative analysis)
  - Machine problems (coding): simulation and kernel hacking

# Exams

- Two exams (midterm & final) @ 20% each:
  - scores may be linearly scaled so median/mean is 75%
  - tentative midterm exam date: **March 10**

# Grade scale

- A:  $\geq 90\%$
- B: 80-89%
- C: 70-79%
- D: 60-69%
- E:  $< 60\%$

# § Class Overview

# CS 450

- Capstone of the systems sequence (CS 350 → 351 → 450)
- Wrap up answer to “how do modern (general purpose) computers work under the hood?”
- The OS is the bedrock of almost all modern software!

You should already know what services are *provided* by OSes, along with:

- how to invoke them (syscalls)
- how to use them effectively and efficiently

lingering questions:

- how are processes actually created/tracked?
- how do processes safely & efficiently share resources (e.g., CPU/Mem)?
- how to correctly/safely leverage concurrency?
- how does the file system work?
- how are protection/security enforced?



# Primary topics

- Kernel architecture
- Processes and Threads
- Scheduling
- Virtual memory
- I/O architectures and device programming
- File Systems
- Interprocess Communication
- Concurrency and Synchronization

# Theory vs. Implementation

- Grand academic debate
- OSes is a huge topic; hard to adequately address both!
  - theory comes first — (hopefully) broad application
  - but it'd be nice to see some working OS code, too ...
- Liberal Arts, Architecture majors have “art history/appreciation” classes
  - Why don't we have “code appreciation”?

*... the best way to prepare [to be a programmer] is to write programs, and to study great programs that other people have written. In my case, I went to the garbage cans at the Computer Science Center and fished out listings of their operating system.*

- Bill Gates

We'll read/tinker with an existing OS codebase, while making modifications and additions

- great way to understand how OSes work without writing millions of lines of code!
- Our OS, xv6, is based on UNIXv6 — released in 1975, among the first preemptive multitasking OSes, and still a great software engineering blueprint!

# Topic for next time: What is an OS?

- Ideas for definition(s):
  - Based on its place in the hardware/software stack
  - Based on its privileges
  - Based on its roles & responsibilities

Before Friday, please read OS:TEP chapters 1 & 2!