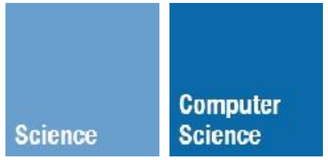
Syllabus & Overview



CS 450 : Operating Systems

Michael Saelee <saelee@iit.edu>

Michael (Sae) Lee

Email: saelee@iit.edu

Office: SB 226C

Hours: Wed & Fri 12:45-13:45

Agenda

- prerequisites
- resources: website, textbooks, etc.
- evaluation: assignments, exams, grading
- class overview
- lecture procedures

§ Prerequisites

CS Essentials

- Procedural programming
- Essential algorithms & runtime analysis
- Data structures
- Data representation

Programming Knowledge

- Languages: Assembly (x86 or other), C/C++
- Compilation process (assembly, compilation, linking, etc.)
- Debugging (with a real debugger!)

Hardware Provisions

- Von Neumann model
- Instruction Set Architectures
- Memory hierarchy (caches thru disk)
- Call stack and stack conventions
- Interrupt Procedures

Operating System Provisions

- the *process*
- exceptional control flow: interrupts, exceptions, and exception handling
- context switches and concurrency
- virtual memory
- input/output

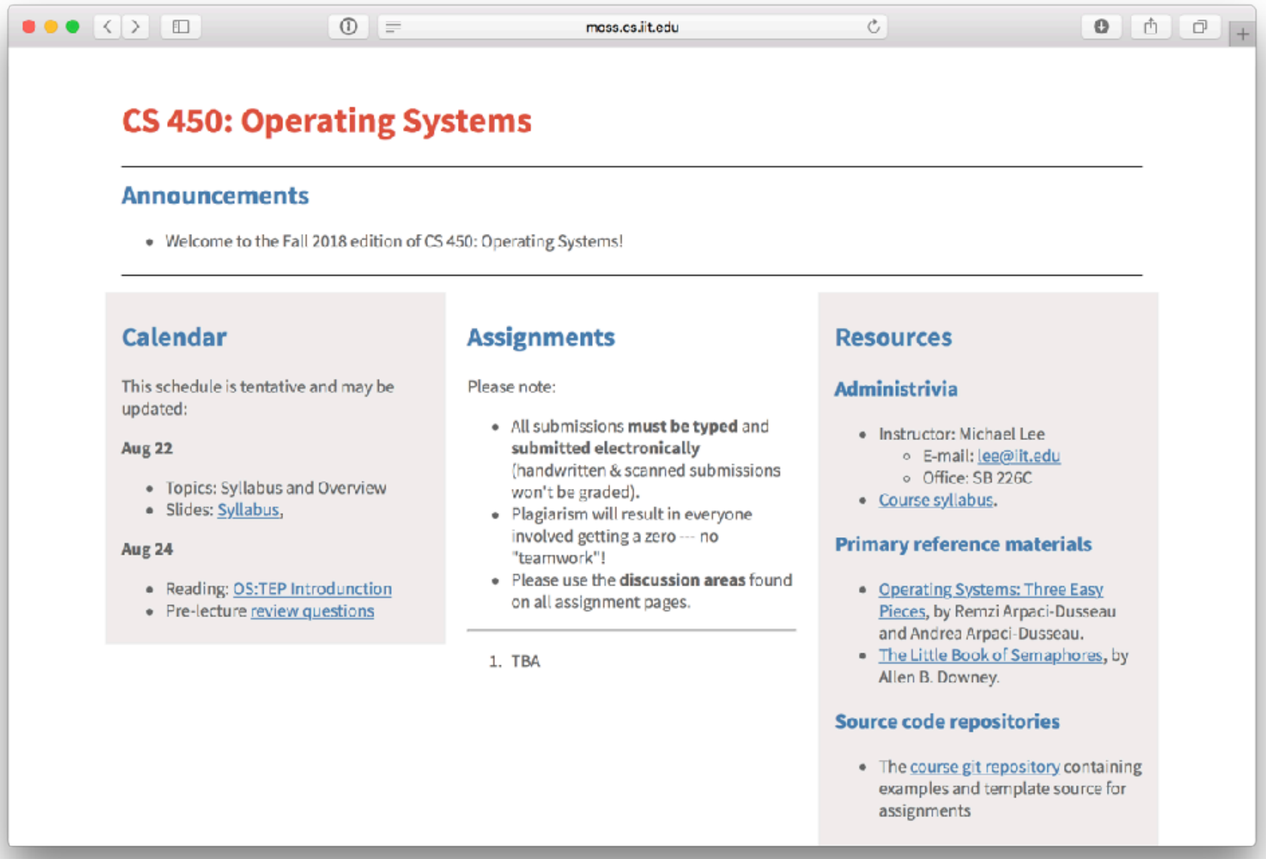
UNIX APIs

- process management (`fork`, `exec`, `wait`, `exit`, etc.)
- exceptional control flow (`kill`, `signal`, etc.)
- input/output (`open`, `close`, `read`, `write`, etc.)

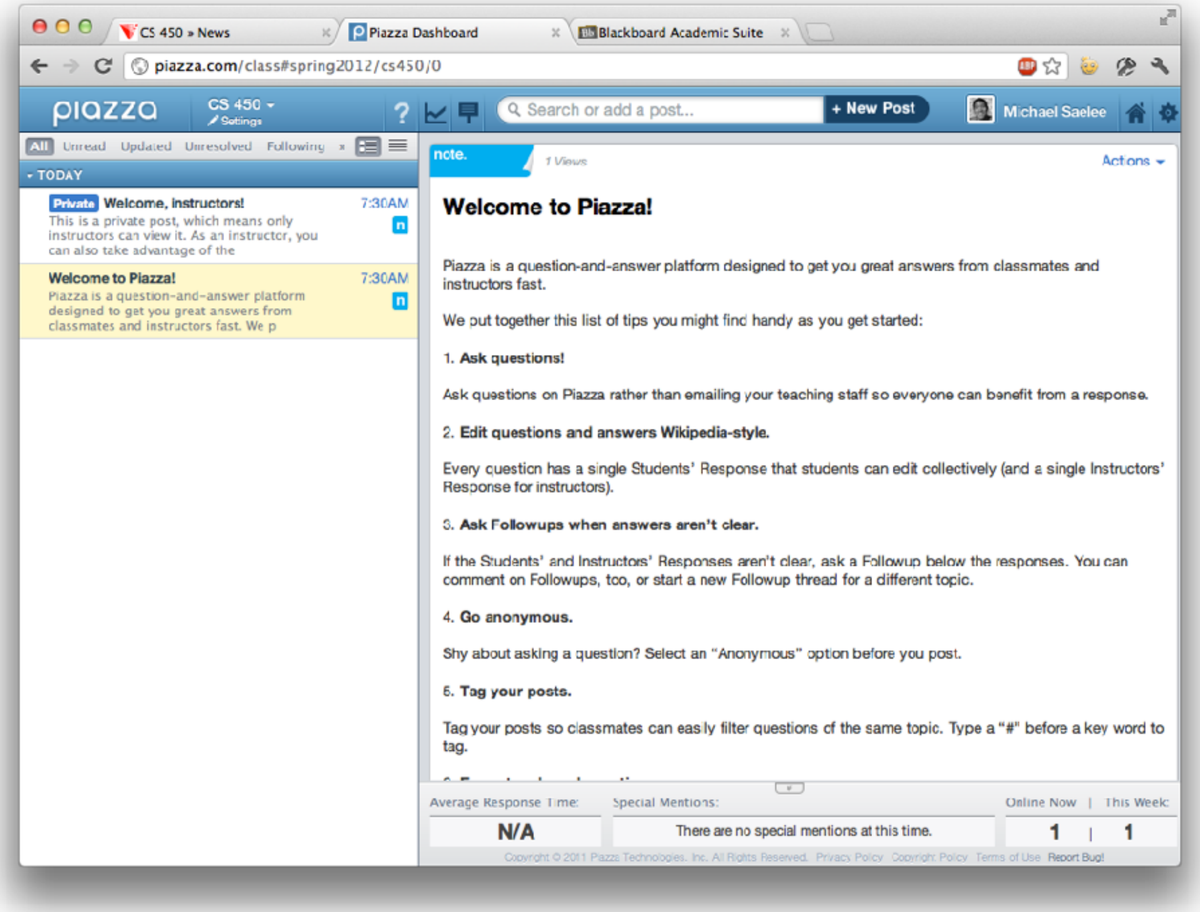
Support Tools

- Debugger (e.g., GDB)
- Build automation (e.g., Make)
- Version control (e.g., Git)
- Virtual Machines, Emulators, Provisioning tools (we'll cover these in this class!)

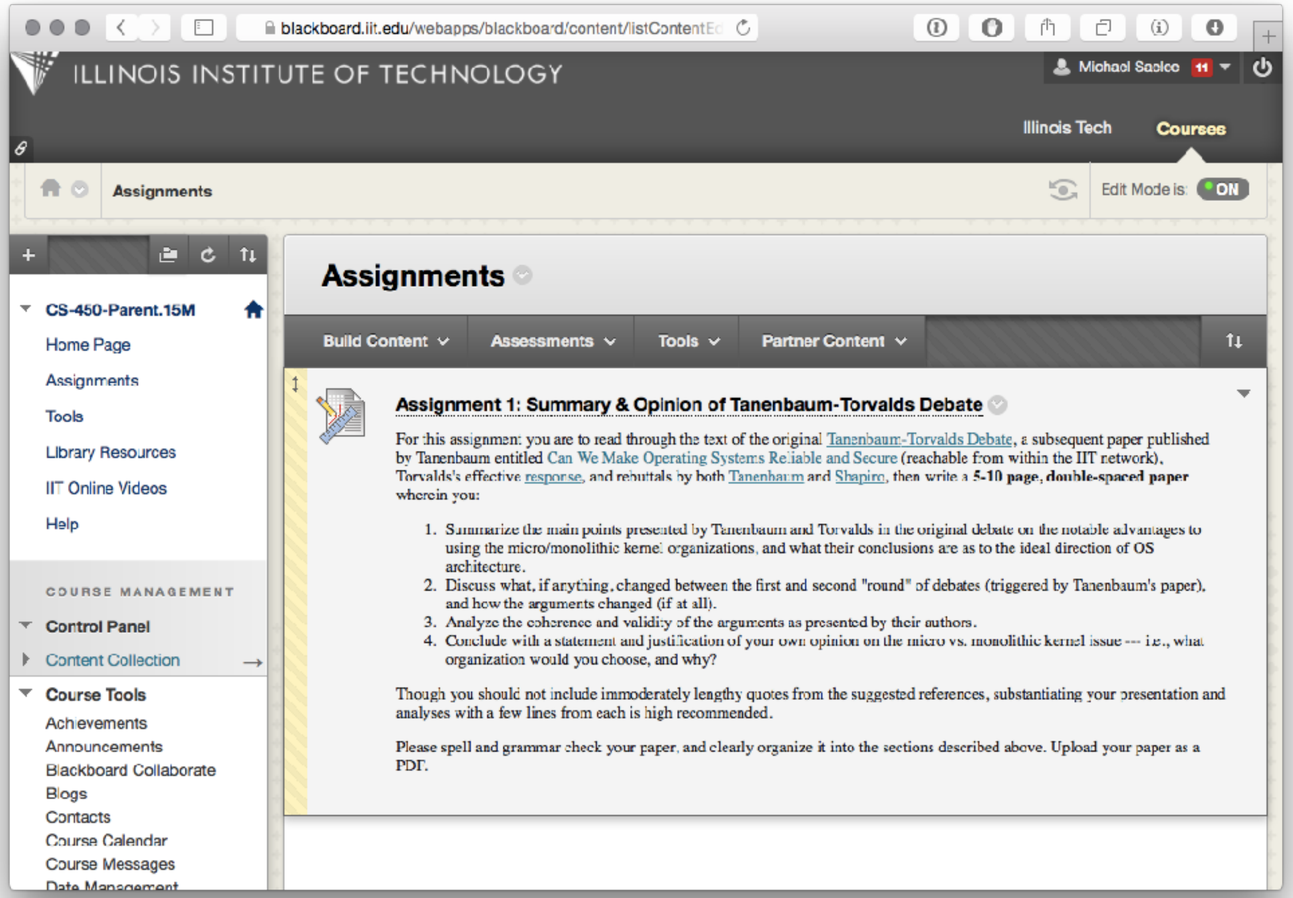
§ Resources



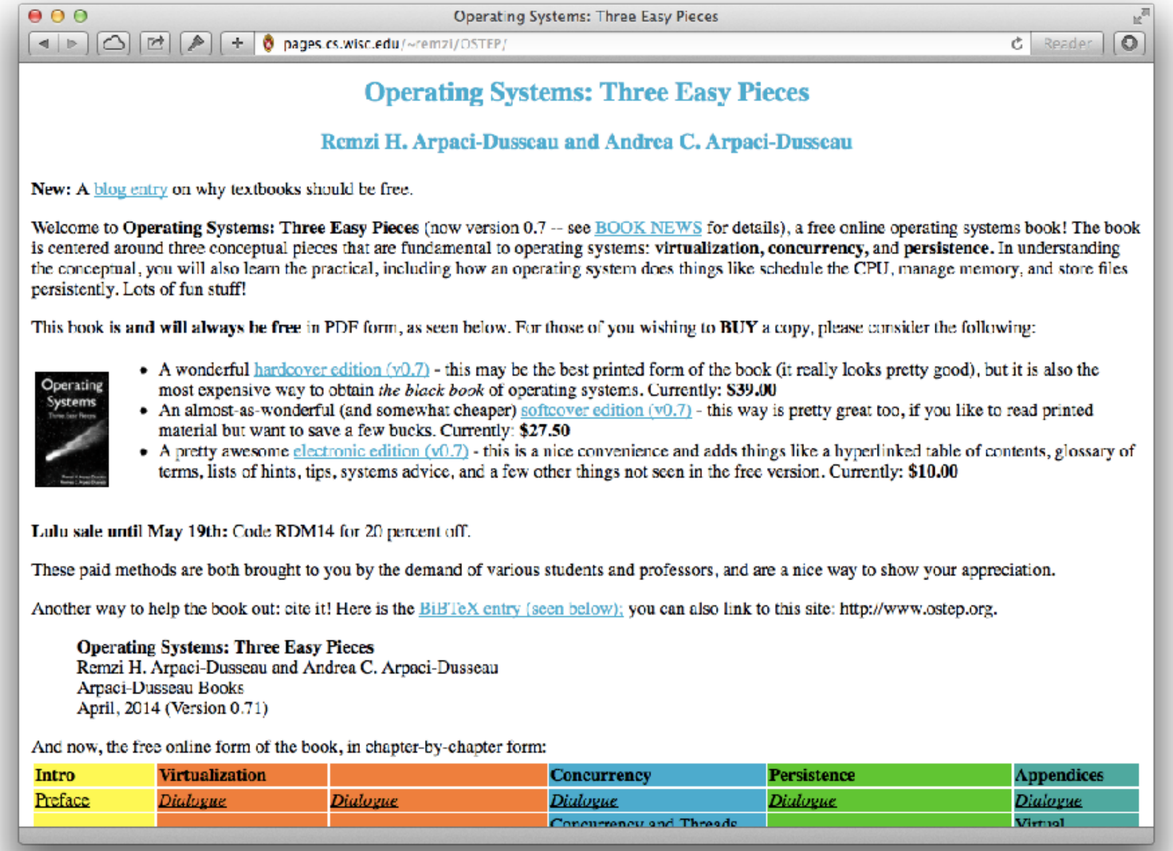
Class website: http://moss.cs.iit.edu/cs450



Class Q/A forum: http://piazza.com



Blackboard: http://blackboard.iit.edu



Textbook: “Operating Systems: Three Easy Pieces”

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

§ Evaluation

5-8 assignments — 45% of grade:

- written papers
- quantitative analyses
- machine problems (coding): simulations and OS implementation

two exams (midterm & final) @ 25% each:

- *no* curving, scores normalized to 75%
  - score ≥ 50% on both exams to pass

remaining 5%: *participation!*

- you start with full credit — each time I call on you (randomly) and you aren't present or don't respond satisfactorily, you lose 1% (excused absences are fine)

A: ≥ 90%

B: 80-89%

C: 70-79%

D: 60-69%

E: < 60%

§ Class Overview

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

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

you should be familiar with details of:

- exceptional control flow
- virtual memory management constructs (e.g., page tables, TLB, etc.)
- basic I/O structures (open file descriptions, FDs, etc.)

lingering questions:

- how are processes scheduled?
- how to correctly/safely leverage concurrency?
- how is the file system implemented
  (and how does I/O work, in general)?
- how are protection/security enforced?

primary topics:

1. scheduling and process management
2. concurrency and synchronization
3. storage management
4. protection and security

plenty of breadth/depth:

- queueing theory
- different approaches to concurrent programming (e.g., message passing)
- OS implementation

the debate: theory vs. implementation

- OSes are too big a topic for both
- theory first – (hopefully) broad application
- but it'd be nice to see some working OS code, too ...

§ Lecture Procedures

Reading/Review Questions

- (Almost) every lecture has assigned reading from OS: TEP
- Pre-lecture questions will be posted on the course website based on the reading or prerequisite material
  - I'll call on people at random — hopefully, answers will help prompt discussions!

Code Review

- 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

Code Review

- In addition to textbook reading, source code will be assigned as reading (with accompanying questions)
  - Reviewed in class — lead by me/students
  - You'll both read and write kernel code as part of MPs

Lecture 02 reading/review questions: (Reading is OS:TEP Introduction)

1. Name some high-level OS implementation goals.
2. What are some of the critical resources managed by the operating system?
3. How is virtualization useful?
4. What are some resources virtualized by the operating system?
5. How is protection useful?
6. What is concurrency? Is concurrency necessarily the same thing as parallel execution?
7. Describe a situation where concurrency can cause problems.
8. What mechanism(s) are used to facilitate the execution of a system call?
9. What does "persistence" refer to (in the context of OSes)?