CS240 - Advanced Topics in Operating Systems

`http://cs240.stanford.edu`

Mendel Rosenblum
mendel@cs.stanford.edu

Today's Agenda - What is CS240?

- What is advanced topics in operating systems?
- Who is teaching this course?
- What is expected of the student?
- How is the course grade computed?
- What systems do we use for course communication?
- Can I use ChatGPT?
- What is the class time going to be like?

Advanced Topics in Operating Systems

- Advanced : Graduate-level course about Operating Systems
  - Assumes you had undergraduate level OS course material (e.g CS111, CS 112 or CS140)
- Topics: Dive deep into topics rather than covering all Operating Systems
  - Concurrency
  - Memory Management
  - File Systems
  - Operating System Structure
  - Systems for Machine Learning
  - History/Experience

Course Organization - All discussion, minimal lecture

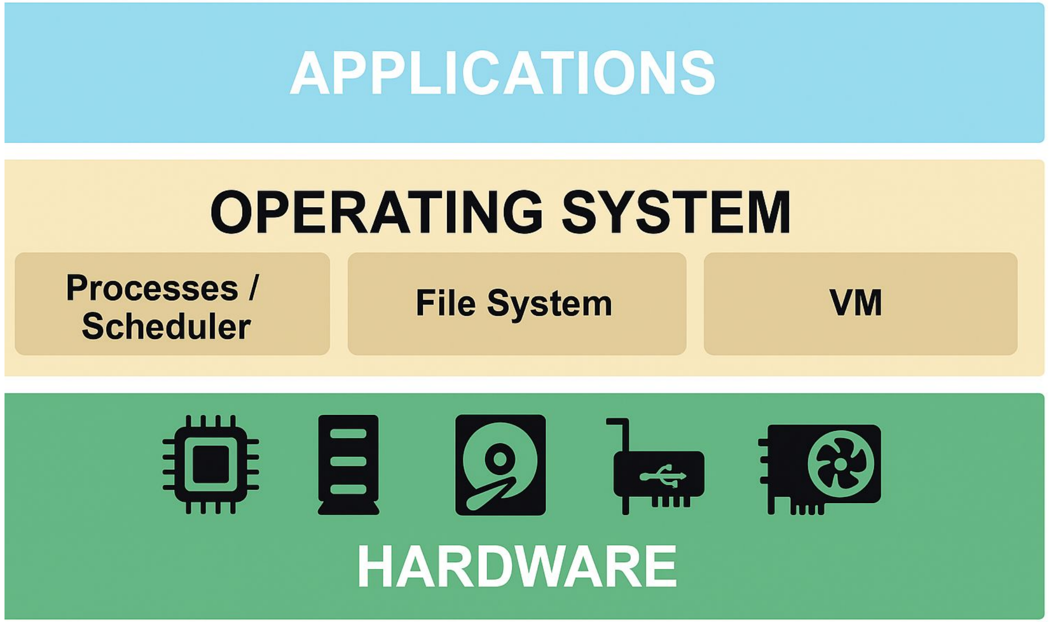
- Each class: a paper or two assigned that you are expected to read before class
  - Need to read paper carefully — assumption is as much as 10–15 hours a week
- Class time (1:30-2:50 PM Tuesday, Thursday) used for paper discussions
  - Won't use lecture time to review paper
  - Participation is factored into grade & attendance is mandatory
    - CGOE/SCPD may attend in-person, but are not required
  - Reading questions submitted on Gradescope on some paper due before class
    - Reading questions good exam question practice
- One (maybe two) programming labs
  - Lab 1 - User Threads Library
- Midterm and Final Exams

Course Staff Introduction

- Instructor: Mendel Rosenblum (mendel@cs.stanford.edu)
  - Office hours: in Gates 450
    - Tuesdays 3PM-4:30PM
    - Thursdays 10:30AM-12PM

- Course Assistant: Michael Paper (mpaper@stanford.edu)
  - Office hours: in CoDa W324
    - Mondays 1:30PM to 3:30PM
    - Wednesdays 9AM to 11AM

What is an Operating System?



- OS - Layer on hardware
  - Manage, abstract, multiplex hardware for applications (i.e. users)
- What is Advanced OS?
  - New abstractions
  - New, better hardware
  - Better management
  - Software engineering

Topic and Paper Selection

Topic selection:

- Classic important OS areas
- Hardware change driven

Papers:

- Read old, classic paper in Operating Systems
  - Classic papers make fewer reader background assumptions
  - Fun to read original work but not necessarily best explained
  - Learn how to write research papers
- Mixed in some newer papers to track state-of-art research
  - Get a sense of interesting research directions

Papers covered this quarter

- Concurrency
  - Mesa (1980), Eraser (1997), Threads/Events (1993-2005), Livelock (1996)
- Memory Management
  - Superpages (2001), ESX (2003),
- Virtualization
  - VMware (2006)
- File Systems
  - NFS (1985), Leases (1989), LFS (1992), Sync (2013), F2FS (2015)
- Operating System Structure
  - Exokernel (1992), Dune (2012)
- Systems for Machine Learning
  - Salus (2019), vLLM (2023)
- History/Experience
  - Hints (1983), TBA

Grade computation

- Exams (60%) - Open book/papers, no electronics
  - Midterm Exam: In class. Tuesday, October 21, 1:30 PM - 2:50 PM
  - Final Exam: Normal final slot. Tuesday, December 9, 3:30 PM - 6:30 PM
- Participation (30%)
  - Class attendance and participation
- Programming Assignments (10%)
  - One or two labs
    - C language, low-level programming

Course Communication

All of these are linked on the course website: http://cs240.stanford.edu

- Course question & discussions: Ed Discussion Forum
  - https://edstem.org/us/courses/86809
  - Prefer mechanism
  - Supports anonymous (to other students) posting
  - Support private communication with course staff
- Reading questions, labs submission, grade distribution: Gradescope
  - https://www.gradescope.com/courses/1129323
- Course Calendar on Canvas
  - https://canvas.stanford.edu/calendar
- Emailing course staff
  - cs240-aut2526-staff@lists.stanford.edu

Generative AI (e.g. ChatGPT) and CS240

- You may not use generative AI for anything submitted for the class
    - Reading questions
    - Programming labs
    - Exams
- Using generative AI in lieu of careful reading of the papers won't turn out well
    - Take care how you use AI

Let's read a paper and discuss it

The Rise of Worse is Better

Richard P. Gabriel

First distributed in 1989, published 1991

10 minutes

The Rise of Worse is Better

- More opinion piece than research paper
    - No references
- Richard P. Gabriel (Dick Gabriel)
    - Stanford Adjunct Professor associated with AI lab.
    - Lucid Software co-founder
- Lisp Machines were advanced, loved by Lisp programmers, but expensive
- Losing to Unix machines, Lucid switched to C++ IDE few years later
- New Jersey?

Some questions

- Describe how the two approaches differ
- Describe what metric for each characteristic

Characteristic	MIT/Stanford vs New Jersey
Simplicity	
Correctness	
Consistency	
Completeness	

What is the PC loser-ing problem?

- 1960s OS from MIT: Incompatible Time Sharing (ITS) OS
  - PDP-10 mainframe computer
  - The right way?

Unix code examples

- Correct way on Unix…

```
ssize_t n;
do {
    n = read(fd, buf, count);
} while (n == -1 && errno == EINTR);
```

- Works almost all the time…

```
err = write(fd, buf, count);
if (err < 0) {
   perror(“write”);
   exit(1)
}
```

```
write: No space left on device
write: Interrupted system call
```

…it takes a tough man to make a tender chicken?

Where did this come from? What does it mean?

https://www.youtube.com/watch?v=In4wh0f3eRA

Other issues

- Community - Is MIT/Stanford or New Jersey better?
- Machine speed - What machine speed does the software assume?
- Monolithic design or component-based design?
- Language of AI: (not C)?

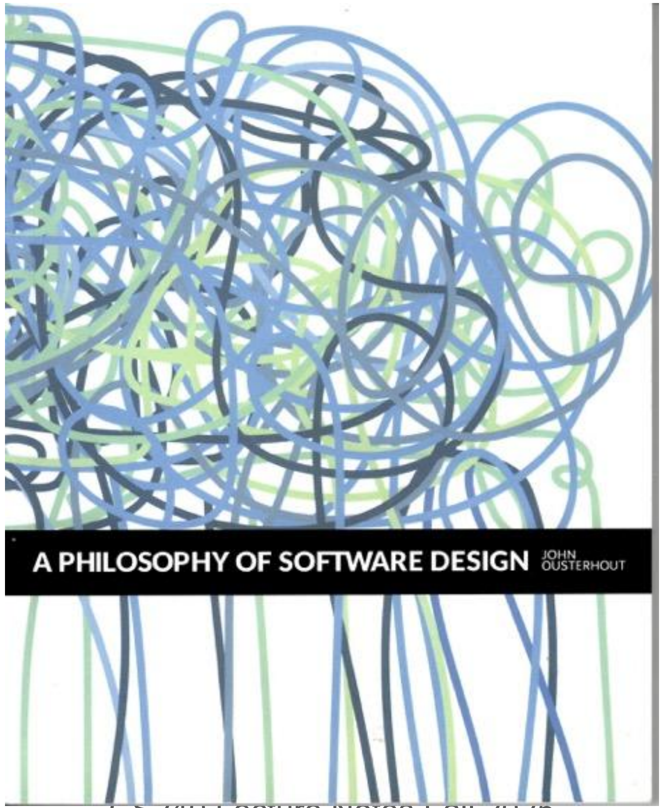
From business schools

- Wikipedia https://en.wikipedia.org › wiki › Minimum_viable_product

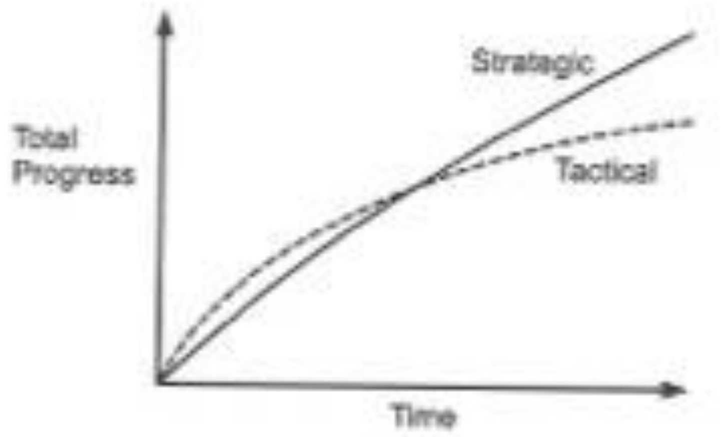
A minimum viable product (MVP) is a version of a product with just enough features to be usable by early customers who can then provide feedback for future.

- Perfection is the enemy of good enough

Let's compare with a 2018 book:



Chapter 3: Strategic vs. Tactical Programming



Tactical example: Facebook

More fast and break things

Strategic examples: Google & VMware

MIT/Stanford or New Jersey?

Chapter 8: Pull Complexity Downwards

It is more important for a module to have a simple interface than a simple implementation.

Is MIT/Stanford or New Jersey?

Chapter 10: Define Errors Out Of Existence

- Exceptions add complexity (uncommon code paths, etc.)

Is MIT/Stanford or New Jersey?

Next class (Thursday, 9/25)

Read: Eraser: A Dynamic Data Race Detector for Multithreaded Programs