CE 440 Introduction to Operating System

Lecture 1: Introduction
Fall 2025

Prof. Yigong Hu

BOSTON UNIVERSITY

Slides courtesy of Manuel Egele, Ryan Huang and Baris Kasikci

Course Instructor

Prof. Yigong Hu

- Assistant Professor, joined in Fall 2025
    - https://yigonghu.github.io/
- Research on Computer System and Software Reliability
- Office: PHO335

Office Hours

- Wed: 3:30-4:30PM (or by appointment)

Staff: Teaching Assistants

William Wang (TA)

- Office Hours: check on course website
- Location: PHO 305/307

Matthew Kweon (grader)

- Office Hours: check on course website
- Location: PHO 305/307

Amado Diallo (grader)

- Office Hours: check on course website
- Location: PHO 305/307

Course Overview

An introductory course to operating systems

- Understand classic OS concepts and principles
- Develop practical system programming skills
- Prepare yourself for advanced distributed and cloud systems

A course with hands-on experience

- Four large programming assignment on a small but real OS
- Work directly with OS internals through practical coding
- Connect course concepts to real implementations

It Will Be a Tough Class

Requires proficiency in systems programming

- Programming in C at a low level
- Concepts are abstract
- Must combine lectures ideas with independent problem-solving

Each projects require significant time

- 20+ hours per assignment
- Be patient and persistence

Why It's Worth It?

- Operating systems are everywhere

Smart phones

↓

iOS

Mobile OS

Laptop

↓

Windows 11

Desktop OS

IoT device

↓

Linux EMBEDDED

Embedded OS

Datacenter

↓

Datacenter OS

Accelerators

↓

cuDNN

ML System

Operating Systems: Then and Now

- ~ 4000 multi/div per second
- 32K 36-bit memory
- $2,630,000+
- Half a room

IBM 709

- Most powerful computer in 1950~

- CPU: 2.0 GHz, 10 cores
- 16G memory
- $1,000
- 13 inch

MacBook Air (2025)

- The cheapest setting

Even Today's cheapest laptop outperforms the most powerful computer of the 1950s

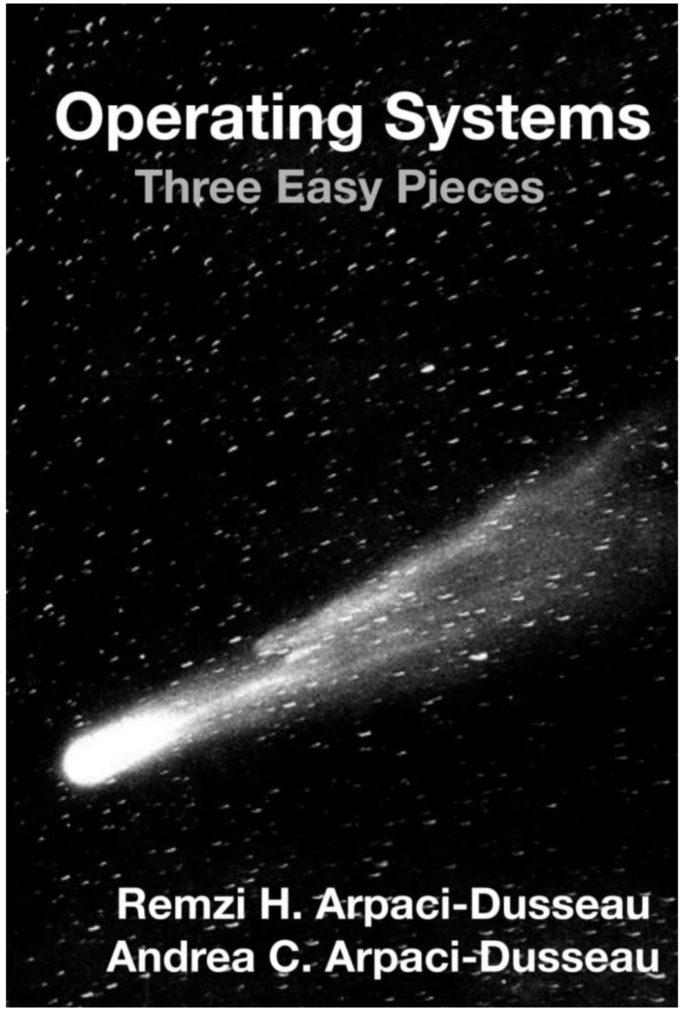
Three Principles in OS

Virtualization	Concurrency	Persistence
Process	Thread	Storage
Virtual Memory	Synchronization	File Systems

Three Fundamental Principles

Textbook

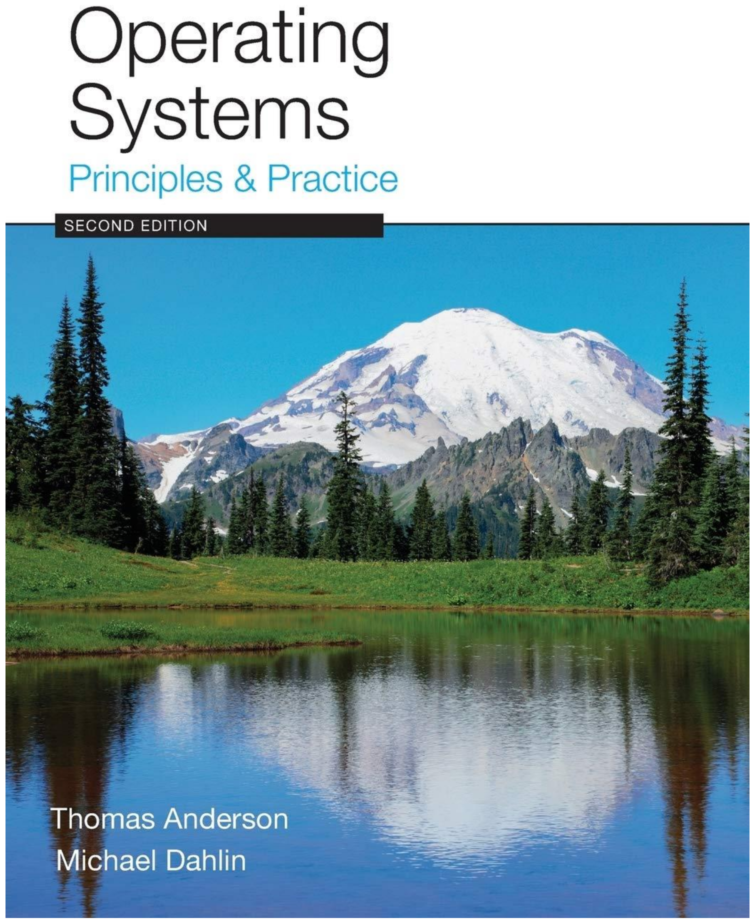
https://pages.cs.wisc.edu/~remzi/OSTEP/

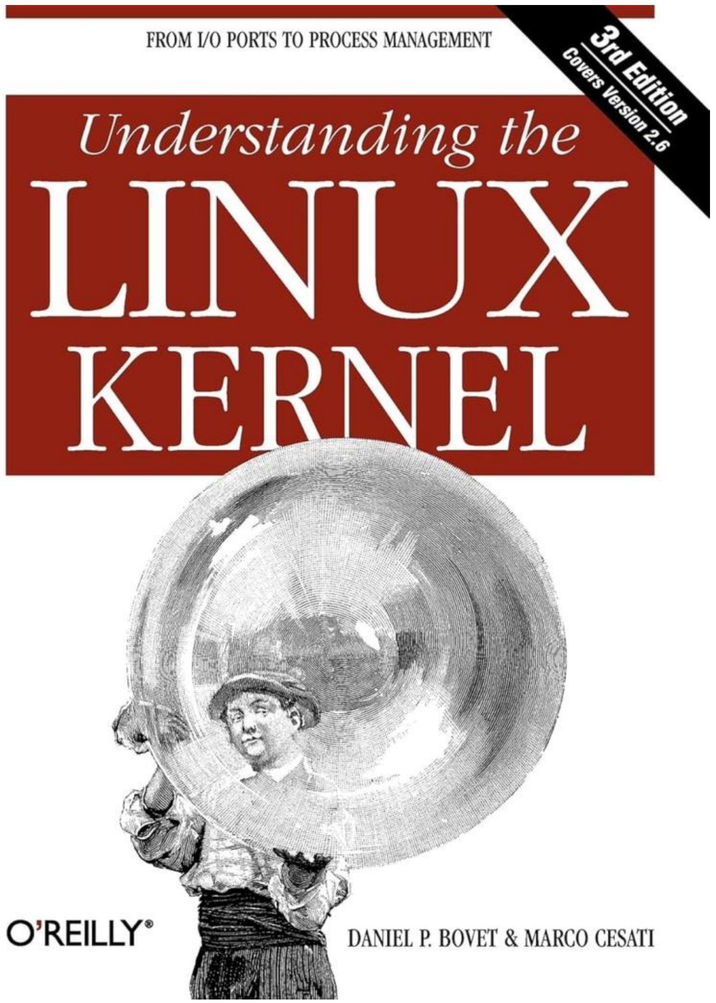


Operating Systems: Three Easy Pieces, Version 1.10

By *Remzi Arpaci-Dusseau* and *Andrea Arpaci-Dusseau*

Other Recommended Textbooks




FROM I/O PORTS TO PROCESS MANAGEMENT
3rd Edition
Covers Version 2.6
Understanding the
LINUX
KERNEL
O'REILLY
DANIEL P. BOVET & MARCO CESATI

Class material

Course website

- https://yigonghu.github.io/EC440/fall25/
- Syllabus, lecture slides, project guidance, homework
- *Check the website weekly*

Piazza

- https://piazza.com/bu/fall2025/ec440
- Ask questions about projects, lectures, and exam
- Use Piazza instead of email for course questions

Grading

Exam: 35% = 15%(Midterm) + 20%(Final)

Projects: 55%

- 3 major labs and 1 warm up lab
- For each project
  - 70% based on passing test cases
  - 30% based on design document

Participation: 10%

- Class Participation

Homework

5 homework assignments throughout the semester

- Optional, no grading
- Help you refreshing the understanding about lectures
- Prepare you for the exams

Exams

Midterm

- Cover first half of class

Final

- Covers the second half of class
- Include project questions

Projects Overview

Implement Pintos operating system

- Developed in 2005 for Standford’s CS 140 OS class
- Written in C, targets x86 hardware
- Runs on a real machine but we will use emulator(QEMU)
- You will implement key OS components (scheduling, memory, syscall)

Pintos is small enough to understand, but real enough to teach core OS concepts

Live Demo: Running Pintos

Projects Assignments

One Setup lab(lab 0)

- Due on Sept 19th (done individually)

Four labs:

- Required: Threads, User processes, Virtual memory
- Bonus: File system: 20%, capped with 100%

Implement projects in groups of up to 3 people

- Find your teammates today

Projects Assignments

Automated tests

- All tests are given so you immediately know how well your code perform
- You either pass a test case or fail, there is no partial credit

Design document

- Answer important questions related to your design for a lab

Coding style

- Make your code easy for TAs and teammates to read

Project Lab Environment

Docker image

- Ubuntu 18.04
- All the environment set up

Virtual Machine

- Download the image on course website

Late Policy

Late submission receives penalties:

- 1 day late, 10% deduction
- 2 days late, 30% deduction
- 3 days late, 60% deduction
- After 4 days, no credit

Each team have a total of 6 days grace period

- Can spread into 4 project
- No question asked
- Use it wisely, strongly suggest to reserve it for lab 2/3

Collaboration Policy

Collaboration

- Explaining a concept to someone in another group
- Discussing algorithms/testing strategies with other groups
- Helping debug someone else’s code (in another group)

Collaboration Policy

Do not look at other people's solution

- Including online solutions
- We will run tools to check for potential cheating

Do not publish your own project

- Online or share with other teams

Cite any code that inspired your code

- If you cite what you used, it won't be treated as cheating
  - In worst case, we only deduct a few points if it undermines the assignment

GPT Policy

GPT are encouraged for

- Debugging your code
- Understanding concepts
- Getting explanations in different styles

You may even try to use GPT to write the entire code

- This will be very challenging
- If you do success in main lab, share your prompt with me — I'll give you **extra bonus**

Honesty matters

- always cite if GPT generated significant code

Tips for passing OS

Come to the lecture

- Lecture is the basis for exams and directly related to projects

Do homework before the exam

- Concepts may seem straightforward…Until you apply them
- Excellent practice for the exams

Start your project early

- The projects cannot be done in the last few days
- Debugging, ask help from TAs and classmates takes time

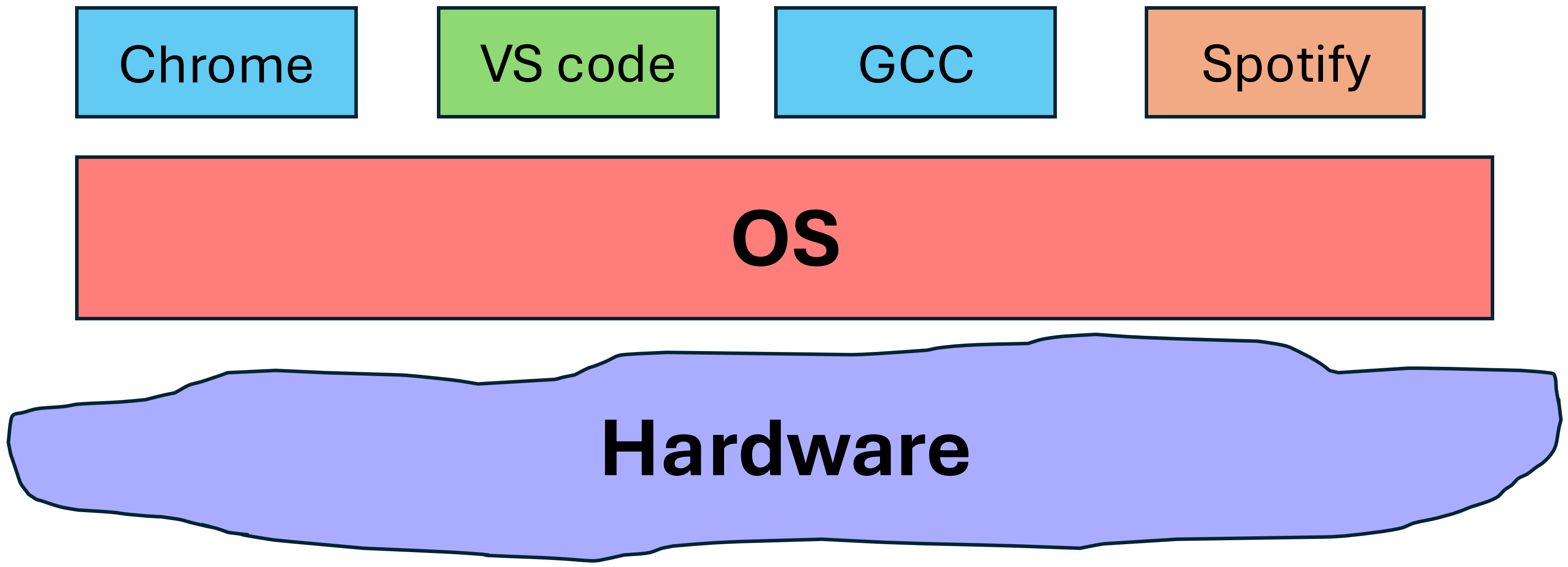
Questions

Any questions?

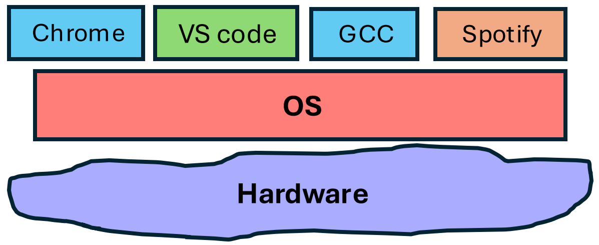
What is an Operating System

An operating system is

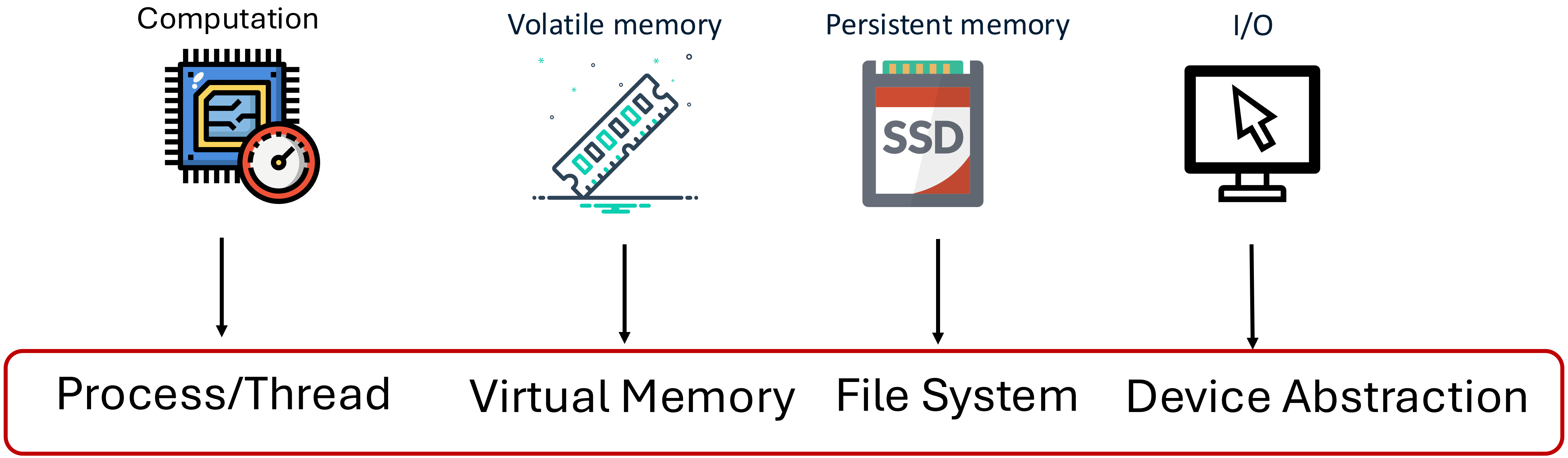
- A software layer between applications and hardware
- “all the code you didn’t write” to implement your application



OS Provides Abstractions

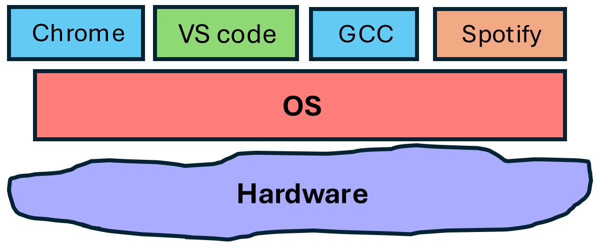


Manage hardware resources:



Abstraction: hide details of hardware and provide clean, uniform interface to application

OS as a Resource Manager



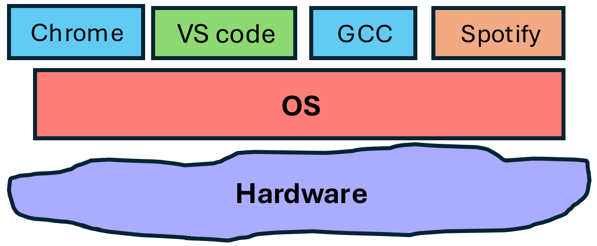
OS manages the hardware resources

- Multiplex: create the illusion of many virtual resources from one physical resource
- Allocate: decide who can use which resource for how much and when

What are the Benefits?

- Simple (no tweaking device registers)
- Device independent (*Same system call for different disks)*
- Portable (across Win95/98/7/8/10)

OS and Applications



OS is main program

- Calls applications as subroutines
- Illusion: every app runs on its own computer

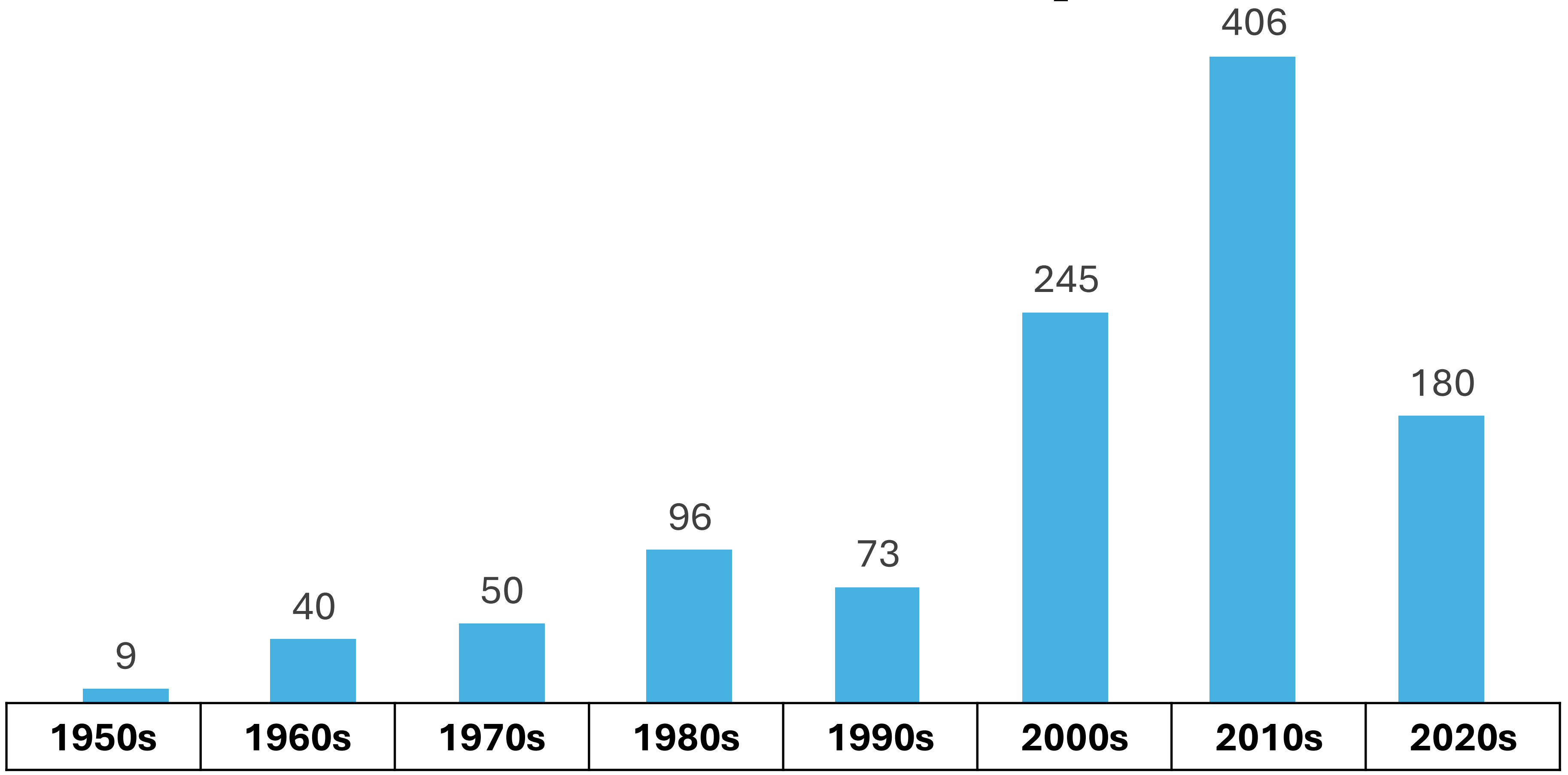
Provide protection

- Prevent one process messing other process

Provide sharing

- Concurrent execution of multiple programs
- Communication among multiple programs
- Shared implementations of common module like file system

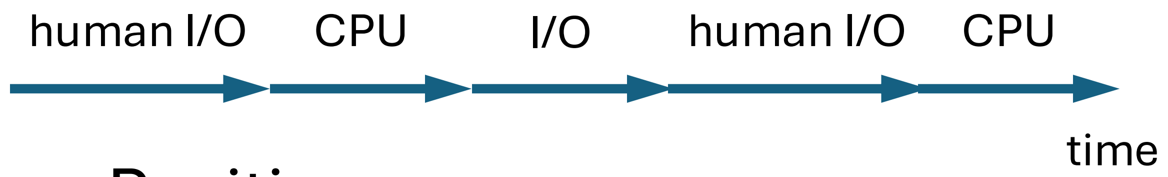
Number of New OSes per Decade



Number of new OS's per decade(Wikipedia)

Evolution of Operating Systems

Single operator at console



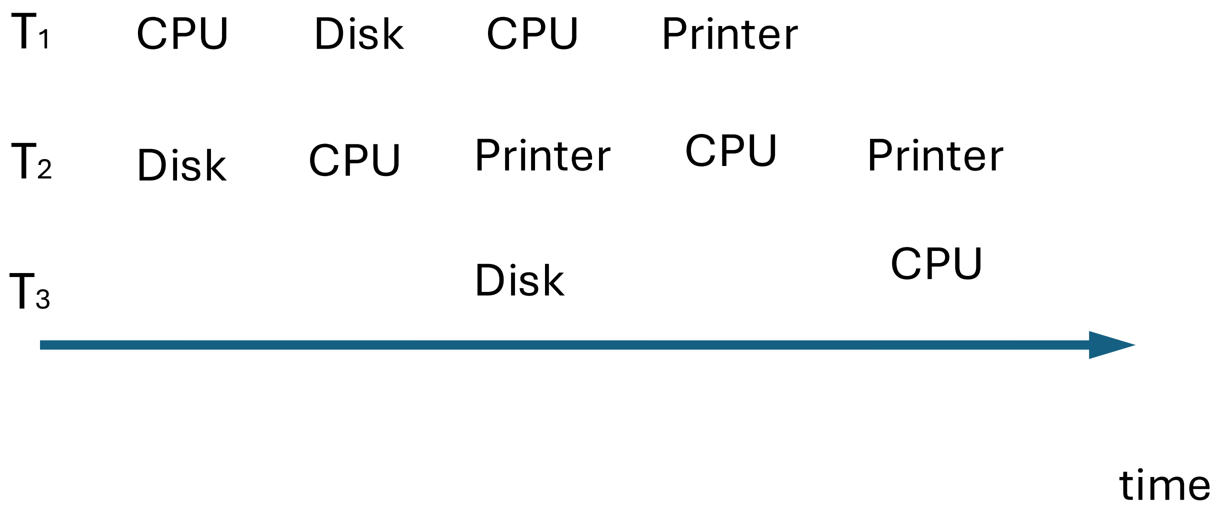
- Positive
  - Interactive
  - Very simple

- Downside
  - Poor utilization

Evolution of Operating Systems

Batch processing

- OS is a batch monitor + library of standard services
- Improve CPU and I/O utilization



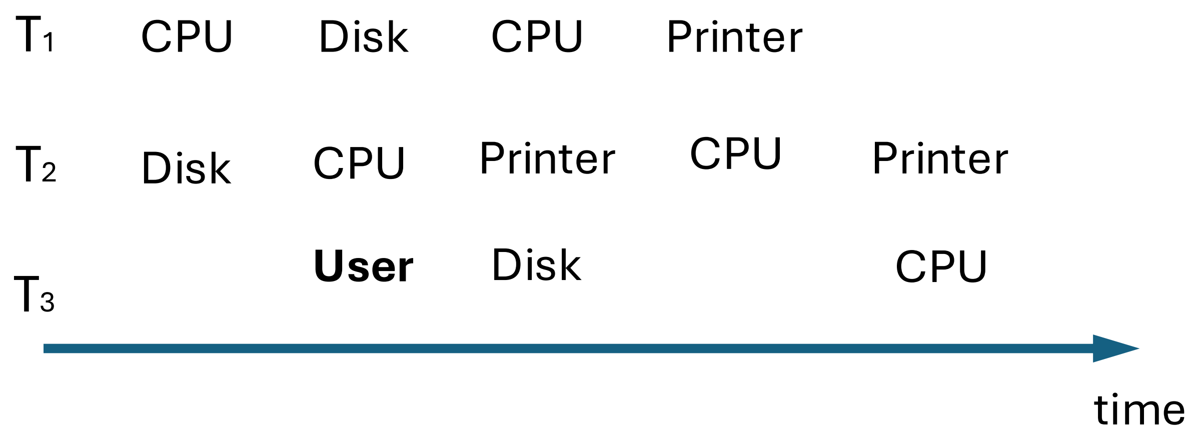
Problem

- Protection becomes an issue
- Not interactive

Evolution of Operating Systems

Time sharing

- Allow people to interact with program as they run
- Insight: User can be modeled as a (very slow) I/O device
- Switch between processes while waiting for user



Evolution of Operating systems

Time sharing

- Allow people to interact with program as they run
- Insight: User can be modeled as a (very slow) I/O device
- Switch between processes while waiting for user

OS is now very complex

Lots of simultaneous jobs

Multiple sources of new jobs(people can start new jobs)

Interactivity is resorted

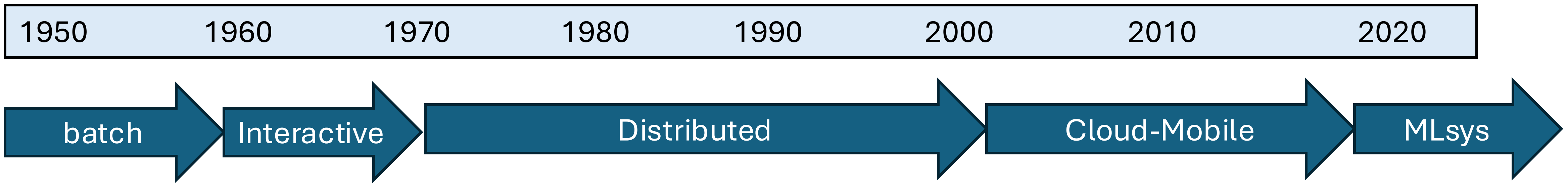
Evolution of Operating systems

OS started out very simple

- Became complex to use hardware efficiently

Consider PCs and workstations

- Is the main assumption (hardware is expensive) still true?



What about today?

Cloud computing

- Amazon EC2
- Is hardware expensive?
- What other OS features are needed?

Mobile computing

- Android/IOS
- What drives efficiency?
- What OS features are needed?

Question to Ponder

OSes continue to evolve

- What are the drivers of OS changes?
  - New hardware, security, new workload

What is part of an OS? What is not?

- Is the windowing system part of an OS?
- Is the Web browser part of an OS?
- OS research has become Dist. Systems research

For Next Class

Browse the course web

- https://yigonghu.github.io/EC440/fall25/

Subscribe to Piazza

Starting Lab0

Start finding partners for project group