Lecture 1 Overview

• Class overview
• Administrative info
• Introduction to operating systems
Personnel

- Instructor
  - Yiying Zhang

- TAs and Tutors
  - Jefferson Chien (TA)
  - Max Gao (TA)
  - Ruohan Hu (TA)
  - Yizhou Shan (TA)
  - Muyuan Chi (TA)
  - Grant Jiang (Tutor)
  - Patrick Lin (Tutor)
  - Joshua Narezo (Tutor)
  - Kyle Wang (Tutor)
  - Fengyuan Wu (Tutor)

- Lectures: T/Th 2-3:20pm (Zoom)
- Discussion session: F 4-4:50pm (Zoom), starts from next week
About Me (https://cseweb.ucsd.edu/~yiying/)

Vincent C. Rideout (MS 1940, Caltech)
-- Gerald Estrin (PhD 1951, University of Wisconsin)
----- David Martin (PhD 1966, University of California at Los Angeles)
------ David Patterson (PhD 1976, University of California at Los Angeles)
------- Remzi and Andrea Arpaci-Dusseau (PhD 1999, University of California Berkeley)
--------- Yiying Zhang (PhD 2013, University of Wisconsin)

• Research interests:
  ◆ Operating systems
  ◆ Distributed systems
  ◆ Computer architecture
  ◆ Systems+networking, systems+security, systems+PL, systems+ML
  ◆ Undergraduate research opportunities at my lab (WukLab)!
CSE 120 Class Overview

• Course material taught through class lectures, textbook readings, and handouts
• Zoom lectures (link available on Canvas)
  ♦ Lectures will be recorded and available on Canvas
  ♦ No class attendance recorded, but strongly encourage all of you to attend Zoom live => actively thinking and asking questions right away is the best and easiest way to learn
• Lecture slides
  ♦ I will post slides in the morning of class day
• Course assignments are
  ♦ Homework questions
  ♦ Two/three large programming projects in groups
  ♦ Midterm and final exams
CSE 120 Class Overview

• Discussion sections
  ♦ TAs will go over projects and answer questions related to projects, homework, and course materials (mainly projects)

• Discussion forum
  ♦ Piazza

• My office hour hours TBD (send me emails before then)
  ♦ Main forum for asking questions related to lecture materials
  ♦ Zoom link available on Canvas
  ♦ When you join the Zoom meeting, you will be in the waiting room. Post your questions through Zoom chat. I will answer them and accept students in the waiting room in FIFO order

• TA and tutor lab hours: TBD (send emails before then)
  ♦ Main forum for asking questions related to projects
Class Web Page

http://cseweb.ucsd.edu/classes/fa21/cse120-a/

- Serves many roles…
  - Course syllabus and schedule (updated over quarter)
  - Lecture slides
  - Homework handouts
  - Project handouts

- Optional material and supplemental readings
  - Entirely for your interest only
Textbook


**FREE**
Homeworks

- There will be 4 homeworks throughout the quarter
  - Reinforce lecture material
- Homeworks provide practice learning the material
  - You get full credit for a technical answer related to the homework question
  - Amount learned from doing homework is proportional to effort
  - Your choice on how much effort
Nachos Project

"This is the planet where nachos rule."
Nachos

• Nachos is an instructional operating system
  ♦ It is a user-level operating system and a machine simulator
    » Not unlike the Java runtime environment
    » Will become more clear very soon
  ♦ Programming environment will be Java on Unix (Linux)
  ♦ The projects will require serious time commitments
    » Waiting until the last minute is not a viable option!

• You will do two or three big projects using Nachos
  ♦ Concurrency and synchronization
  ♦ System calls, processes, multiprogramming
  ♦ Virtual memory (optional)

• You will work in groups of 1-4 on the projects
  ♦ Start thinking about partners
Labs

• You will need to use your home machine (laptop/desktop) to connect to ieng6
  ♦ You can also try to set up local environments and run locally (we will only provide limited support/help for that)
  ♦ Note: We will test and grade on ieng6 machines
  ♦ Be sure to test your projects there
    » You will be able to test before the deadline

• Lab hours: main forum to ask questions about projects
  ♦ We will use a hybrid form, supporting both in-person and remote
  ♦ Find schedule and Zoom link on course website
  ♦ TAs and tutors will use autograder to order questions (FCFS)
  ♦ They will not debug for you!
  ♦ Everyone gets only up to 10 min => need to serve a big class
Exams

• Midterm
  ♦ Tuesday Oct 26th (reply Piazza poll for your availability by the end of next Friday!)
  ♦ Covers first half of class

• Final
  ♦ Thursday Dec 9th (3pm-6pm) (decided by univ, no change)
  ♦ Covers second half of class

• Both exams will be conducted online through Canvas
  ♦ Open book, you can use any resources
  ♦ Questions will be randomized => no way for you to exchange answers with other students

• No makeup exams
  ♦ Everyone must be able to attend these exam dates
    » Unless absolute dire circumstances
Grading

- If you choose to do only the first two projects:
  - Homeworks: 6%
  - Midterm: 30%
  - Final: 30%
  - Projects: 34%

- If you choose to do all the three projects:
  - Homeworks: 6%
  - Midterm: 25%
  - Final: 25%
  - Projects: 44%
How *Not* To Pass CSE 120

- Do not watch Zoom lecture videos
  - Lecture videos are too long, the slides are online, and the material is in the book anyway
  - Lecture material is the basis for exams and directly relates to the projects. We will cover materials only in the lecture recordings (not in slides or textbooks) in exams!
  - Batch many lectures and watch once, or worse, watch all lectures right before exams!!
  - You will not be able to digest everything! This has been shown to be the way to do poor at exams (and projects)!
  - No class attendance recorded, but strongly encourage all of you to attend Zoom live => actively thinking and asking questions right away are the best and easiest way to learn
How *Not* To Pass CSE 120

- Wait until the last couple of days to start a project
  - We’ll have to do the crunch anyways, why do it early?
  - The projects cannot be done in the last few days
  - Repeat: The projects cannot be done in the last few days

- Do not do the homework
  - It’s only 6% of the grade, get full credit for turning anything in
  - Concepts seem straightforward…until you apply them
  - Excellent practice for the exams, and some homework problems are exercises for helping with the project
Project 1 Scores

![Graph showing Project 1 Scores with start dates from DAY.0-2 to DAY.14-16, and score categories from 75 to 100. The graph compares mean and median scores over time.]
How *Not To Pass Even More*

- Do not ask questions in lecture, office/lab hours, or online
  - It’s scary, I don’t want to embarrass myself
  - Asking questions is the best way to clarify lecture material at the time it is being presented
  - Office hours and email will help with homework, projects

- Violate academic integrity
  - It is much better to get a 0 for an assignment than to fail the course for academic integrity violations
Academic Integrity

• Exams
  - Work them on your own!

• Projects
  - Each team must write their own solution
  - No discussion of or sharing of specific code or written answers is allowed
  - Any sources used outside of textbook/handouts/lectures must be explicitly acknowledged
  - Your responsibility to protect your files from
    » e-copying using UNIX file protection
    » public access, including disposal

• We take cheating very seriously, with a zero tolerance policy
  - We will run tools to catch that, do not even attempt!
Diversity and Inclusion

We are committed to fostering a learning environment for this course that supports a diversity of thoughts, perspectives and experiences, and respects your identities (including race, ethnicity, heritage, gender, sex, class, sexuality, religion, ability, age, educational background, etc.). Our goal is to create a diverse and inclusive learning environment where all students feel comfortable and can thrive.

Our instructional staff will make a concerted effort to be welcoming and inclusive to the wide diversity of students in this course. If there is a way we can make you feel more included please let one of the course staff know, either in person, via email/discussion board, or even in a note under the door. Our learning about diverse perspectives and identities is an ongoing process, and we welcome your perspectives and input.

We also expect that you, as a student in this course, will honor and respect your classmates, abiding by the UCSD Principles of Community (https://ucsd.edu/about/principles.html). Please understand that others’ backgrounds, perspectives and experiences may be different than your own, and help us to build an environment where everyone is respected and feels comfortable. If you experience any sort of harassment or discrimination, please contact the instructor as soon as possible. If you prefer to speak with someone outside of the course, please contact the Office of Prevention of Harassment and Discrimination: https://ophd.ucsd.edu/.
We aim to create an environment in which all students can succeed in this course. If you have a disability, please contact the Office for Students with Disability (OSD), which is located in University Center 202 behind Center Hall, to discuss appropriate accommodations right away. We will work to provide you with the accommodations you need, but you must first provide a current Authorization for Accommodation (AFA) letter issued by the OSD. You are required to present their AFA letters to Faculty (please make arrangements to contact me privately) and to the OSD Liaison in the department in advance so that accommodations may be arranged.
Basic Needs Resources

• Are you eating properly? Do you have adequate access to nutritious food? Do you have stable housing? Are you homeless or couch surfing?

• If you or someone you know has food and/or housing insecurity, please note: http://basicneeds.ucsd.edu
  ♦ The Triton Food Pantry (in the old Student Center), https://www.facebook.com/tritonfoodpantry/, is free and anonymous, and includes produce.
  ♦ Financial aid resources, the possibility of emergency grant funding, and off-campus housing referral resources are available.
  ♦ CAPS and college deans can connect students to the above resources, as well as other community resources and support.
Questions

• Before we start the material, any questions about the class structure, contents, etc.?
Why?

You have a question, Calvin?

Yes! What assurance do I have that this education is adequately preparing me for the 21st century?

Am I getting the skills I'll need to effectively compete in a tough, global economy? I want a high-paying job when I get out of here! I want opportunity!
Why Operating Systems?

• Why are we making you sit here today, having to suffer through a core course in operating systems?
  ♦ It’s not like everyone will become OS developers, after all

• Understand what you use
  ♦ Understanding how an OS works helps you develop apps
  ♦ OS is the foundation of virtualization, what cloud runs on
  ♦ System functionality, performance, efficiency, etc.

• Pervasive abstractions
  ♦ Concurrency: Threads and synchronization are common modern programming abstractions (Java, C#, C++, Go, etc.)

• Complex software systems
  ♦ Many of you will go on to work on large software projects
  ♦ OSes serve as examples of complex systems
This course addresses classic OS concepts
- Services provided by the OS
- OS implementation on modern hardware
- Interaction of hardware and software
- Techniques for implementing software systems that are
  - Large and complex
  - Long-lived and evolving
  - Concurrent
  - Performance-critical

System software tends to be mysterious
- Can your program allocate more memory than what your machine has physically? Why?

Our goal is to explain those mysteries
What this course is not about

- How to use an OS
- Graphic user interfaces of OS’s
- A particular OS (although we use UNIX/Linux to explain many concepts)
- Different OS kernel architectures or distributed OS’s (take grad OS for that)
What is an Operating System?

• How would you answer?
  ♦ (Yes, I know that’s why you’re taking the course…)
  ♦ (Note: There are many answers…)

CSE 120 – Lecture 1 – Course Intro
A Typical Computer from a Hardware Point of View

- CPU
- Chipset
- Memory
- I/O bus
- Network

CSE 120 – Lecture 1 – Course Intro
Computer System Components

user 1
user 2
user 3
... 
user n

compiler
assembler
text editor
... 
database system

system and application programs

computer hardware

?
Computer System Components

User 1 -> Compiler
User 2 -> Assembler
User 3 -> Text Editor
... -> Database System

System and Application Programs

Operating System

Computer Hardware
What is an Operating System?

“Code” that *sits between*:

- programs & hardware
- different programs
- different users

But what does it do/achieve?
What is an OS?

- Resource manager
- Extended (abstract) machine

Makes computers *efficient, easy, and safe* to use

- (will have a 3rd def based on pragmatics next time)
OS Manages Hardware Resources (answer 1)

• The OS controls/mediates/manages access to hardware resources
  ♦ Computation (CPUs)
  ♦ Volatile storage (memory) and persistent storage (disk, etc.)
  ♦ Communication (network adapter)
  ♦ Input/output devices (keyboard, display, mouse, etc.)
OS Manages Hardware Resources (answer 1)

- Allocation
- Reclamation
- Protection

Finite resources
Competing demands
OS Manages Hardware Resources (answer 1)

• Allocation
• Reclamation
• Protection

“The OS giveth
The OS taketh away”

Implied at termination
Involuntary at run time
Cooperative (yield cpu)
OS Manages Hardware Resources (answer 1)

- Allocation
- Reclamation
- Protection

“You can’t hurt me
I can’t hurt you”

Implies some degree of safety & security
• The OS defines a set of logical resources *(objects)* and a set of well-defined operations on those objects *(interfaces)*
  - Physical resources (CPU and memory)
  - Logical resources (files, programs, names)
  - Sounds like OO…

• The **logical, well-defined abstraction** OS provides is much more ideal than the hardware interface
  - Ease to use (no need to deal with low-level interface, hardware registers, different device models etc.)
  - Fair (well-behaved)
  - Supporting backward-compatibility
  - Reliable
  - Secure
Abstraction for Applications
(Answer 2)

• Users and programs can safely coexist, cooperate, and share resources
• with the illusion of infinite, private (reliable, secure) resources
  ♦ Concurrent execution of multiple programs (time sharing)
  ♦ Communication among multiple programs (pipes, cut & paste)
  ♦ Shared implementations of common facilities
    » No need to implement the file system more than once
    » Any down side of this??
  ♦ Mechanisms and policies to manage/share/protect resources
    » File permissions (mechanism) and groups (policies)
How to design an OS?
Is there a perfect OS?

- Efficiency
- Fairness
- Portability
- Interfaces
- Security
- Robustness

• Conflicting goals
  ◆ Fairness vs efficiency
  ◆ Efficiency vs portability
  ◆ …

• Furthermore, …
**Hardware is evolving...**

- **60’s-70’s - Mainframes**
  - Rise of IBM

- **70’s - 80’s – Minicomputers**
  - Rise of Digital Equipment

- **80’s - 90’s – PCs**
  - Rise of Intel, Microsoft

- **90’s - 00’s – handheld/portable systems (laptops)**

- **2007 - today -- mobile systems (smartphones), Internet of Things, specialized hardware in the cloud**
  - Rise of iPhone, Android
## Implications on OS Design Goals: Historical Comparison

<table>
<thead>
<tr>
<th></th>
<th>Mainframe</th>
<th>Mini</th>
<th>Micro/Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System $/worker</strong></td>
<td>10:1 – 100:1</td>
<td>10:1 – 1:1</td>
<td>1:10-1:100</td>
</tr>
<tr>
<td><strong>Performance goal</strong></td>
<td>System utilization</td>
<td>Overall cost</td>
<td>Worker productivity</td>
</tr>
<tr>
<td><strong>Functionality goal</strong></td>
<td>Maximize utilization</td>
<td>Features</td>
<td>Ease of Use</td>
</tr>
</tbody>
</table>
Hardware is evolving (cont) ...

- (once) New architectures
  - Multiprocessors
  - 32-bit vs. 64-bit
  - Multi-core

- New memory, storage, network devices
  - SSD, Optane, RDMA, SmartNIC, programmable switches

- New processors
  - FPGA, GPU, TPU, IPU
May You Live in Interesting Times...

- Processor density (no longer) doubles every 2 years
- Network bandwidth and data rates double every 18 months

→ Performance/cost “sweet spot” constantly changing

* Does human productivity ever double?
Applications are also evolving...

- New applications
  - Computer games, networked games
  - Virtual reality
  - Web 2.0 (search, youtube, social network, …)
  - Video streaming
  - Mobile apps (> 2.8 million iPhone, Android apps)
  - Big data
  - Machine learning, deep learning, reinforcement learning
  - Autonomous vehicles
  - …
Implications to OS Design

- Constant evolution of hardware and applications continuously reshape
  - OS design goals (performance vs. functionality)
  - OS design performance/cost tradeoffs

- Any magic bullet to good OS design?
no magic in OS design

This is Engineering

- Imperfection
- Tradeoffs (perf/func/security)
- Different Goals
- Constraints
  - hardware, cost, time, power
- Optimizations

Nothing’s Permanent

- High rate of change
  - Hardware
  - Applications
- Cost / benefit analyses
- One good news:
  - Inertia of a few design principles
Separating Policies from Mechanisms

A fundamental design principle in Computer Science

Mechanism – tool/implementation to achieve some effect

Policy – decisions on what effect should be achieved

Example – CPU scheduling:
- All users treated equally
- All program instances treated equally
- Preferred users treated better

Separation leads to flexibility!
About this course...

Principles of OS design
• Some theory
• Some rational
• Lots of practice

Goals
• Understand OS design decisions
• Last piece of the “puzzle”
• Basis for future learning

To achieve the goals:
• Learn concepts in class
• Get hands “dirty” in labs
Topics we’ll cover

• Process management
• Memory management
• I/O management (file system)
• A touch of advanced topics if we have time
**Fundamental OS Issues**

- The fundamental issues/questions in this course are:
  - **Structure:** how is an operating system organized?
  - **Sharing:** how are resources shared among users?
  - **Naming:** how are resources named (by users and programs)?
  - **Protection:** how are users/programs protected from each other?
  - **Reliability and fault tolerance:** how to mask failures?
  - **Concurrency:** how to control parallel activities?
  - **Performance:** how to make efficient use of resources, reduce OS overhead?
  - **Scale and growth:** how to handle increased demand?
  - **Compatibility:** can we ever do anything new?
  - **Distribution:** how to coordinate remote operations?

- And the **principles** in this course are the design **methods**, **approaches**, and **solutions** to these issues
Expect (some) pain

Somewhat fast pace

Lots of programming and debugging

Some difficult (abstract) concepts
For next class...

- Browse the course web
  https://cseweb.ucsd.edu/classes/fa21/cse120-a/
- Sign up on Piazza!
- Read Chapters 1, 2, and 6
- Start thinking about partners for project groups
- Let the fun begin!
Backup Slides
Brief History of OS design

In the beginning…

• OSes were runtime libraries
  ♦ The OS was just code you linked with your program and loaded into the computer
  ♦ First computer interface was switches and lights, then punched tape and cards

• Batch systems were next
  ♦ OS was permanently stored in primary memory
  ♦ It loaded a single job (card reader, mag tape) into memory
  ♦ Executed job, created output (line printer)
  ♦ Loaded the next job, repeat…
  ♦ Card readers, line printers were slow, and CPU was idle while they were being used
Spooling

- Disks provided a much faster I/O device than card readers, mag tape, and line printers
- Motivated development of spooling (Simultaneous Peripheral Operation On-Line)
  - Use disk to overlap I/O of one job with computation of others
  - Move program/data from card reader onto disk while another job computes
  - When done, next job ready to be loaded from disk
  - Can spool multiple programs onto disk, OS can choose which job to run next (job scheduling)
  - But, CPU still idle when job reads/writes to disk
Multiprogramming

- Multiprogramming increased system utilization
  - Keeps multiple runnable jobs loaded in memory
  - Overlaps I/O processing of a job with computation of another
  - Benefits from I/O devices that can operate asynchronously
  - Requires the use of interrupts and DMA
  - Optimizes system throughput (number of jobs finished in a given amount of time) at the cost of response time (time until a particular job finishes)
Timesharing

- Timesharing supports interactive use of computer by multiple users
  - Terminals give the illusion that each user has own machine
  - Optimizes response time (time to respond to an event like a keystroke) at the cost of throughput
  - Based on timeslicing – dividing CPU time among the users
  - Enabled new class of applications – interactive!
  - Users now interact with viewers, editors, debuggers

- The MIT Multics system (mid-late 60s) was an early, aggressive timesharing system

- Unix and Windows are also timesharing systems…
Distributed Operating Systems

• Distributed systems facilitate use of geographically distributed resources
  ♦ Machine connected by wires
• Supports communication between parts of a job or different jobs on different machines
  ♦ Interprocess communication
• Sharing of distributed resources, hardware, and software
  ♦ Exploit remote resources
• Enables parallelism, but speedup is not the goal
  ♦ Goal is communication
Parallel Operating Systems

- Support parallel applications trying to get speedup of computationally complex tasks across multiple CPUs
- Requires basic primitives for dividing single task into multiple parallel activities
- Supports efficient communication among activities
- Supports synchronization of activities to coordinate data sharing
- Early parallel systems used dedicated networks and custom CPUs, now common to use networks of high-performance PC/workstations
Embedded Operating Systems

• Decreased cost of processing makes computers ubiquitous
  ♦ Your car has dozens of computers in it
  ♦ Think of everything that has electric motor in it, and now imagine that it also has a computer

• Each embedded application needs its own OS
  ♦ Cell phones
  ♦ PDAs (PalmPilot, etc.)

• Very soon
  ♦ Your house will have 100s of embedded computers in it
  ♦ Your electrical lines and airwaves will serve as the network
  ♦ All devices will interact as a distributed system