

CS162
Operating Systems and
Systems Programming
Lecture 1

What is an Operating System?

Professor Natacha Crooks
<https://cs162.org/>

Intros - Natacha Crooks

Assistant Professor in EECS

- 723 Soda Hall (Sky Lab), <http://nacrooks.github.io>

Member of Sky Lab

Research areas:

- Large Scale Distributed Systems And Databases
- Decentralized Systems

Outside Activities

- Consultant for Blockchain companies and Data Orchestration Companies.

Intros - CS162's Mighty TAs



Wilson Nguyen (Head TA)



Sohom Paul (Head TA)



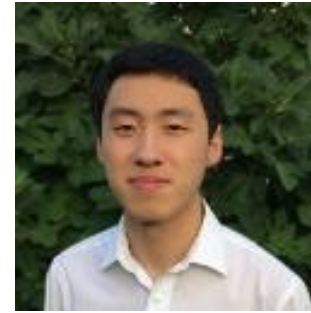
Carlos Huerta



Shreyas Kallingal



Claire Wen



Alvin Xu



Micah Murray



Ryan Alameddine



Tiffany Wang

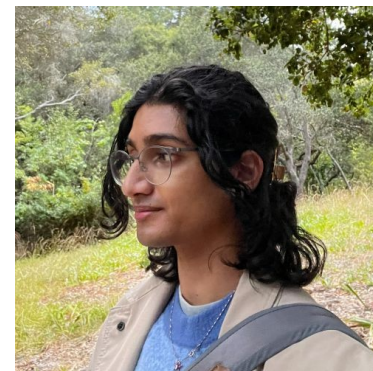
Intros - CS162's Mighty Readers



Ashwin Chugh



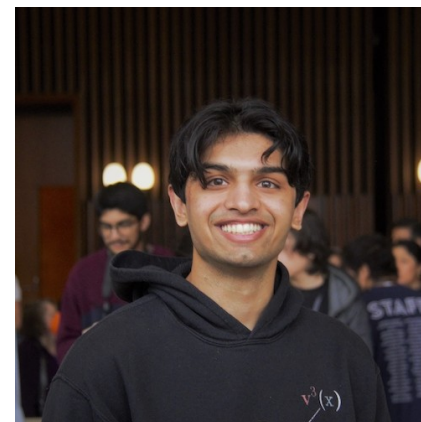
Duc Nguyen



Gaurav Bhatnagar



Edrees Saied



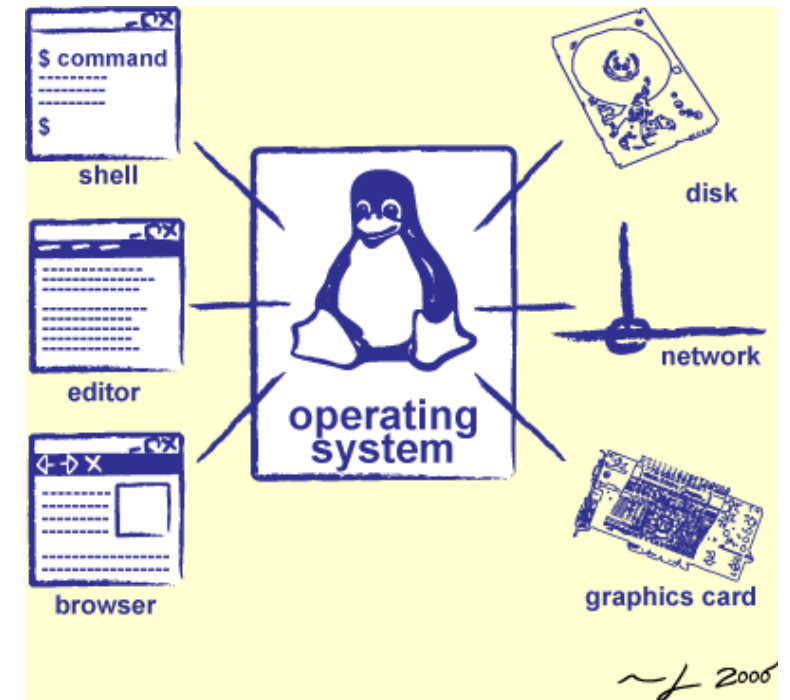
Vivek Verma

We don't bite!



Goals for Today

- Why should you care?
- Why is it hard?
- What is an Operating System?
- Administratrivia & Course Policy



Goal 1: Why should you care?

The OS is everywhere

Every **device**, from your smartwatch, your smart light bulb, to your mobile phone and laptop runs an operating system

Every **program** you will ever write will run on an operating system

Its **performance** and **execution** behaviour will depend on the operating system

Goal 2: Why is designing an OS hard?

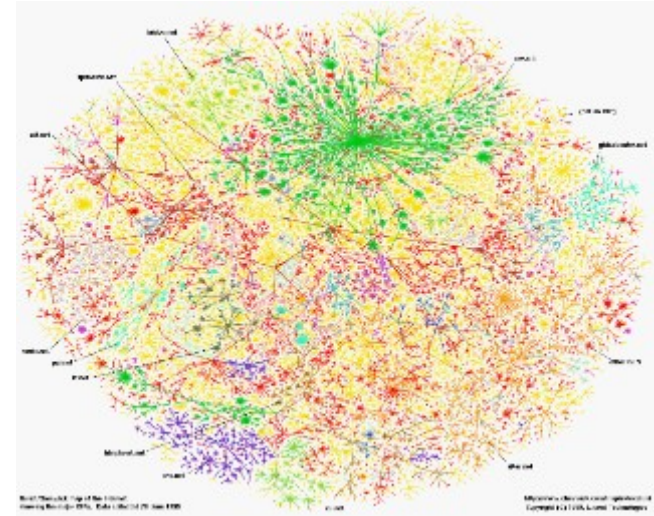
What do these have in common?



Across many devices



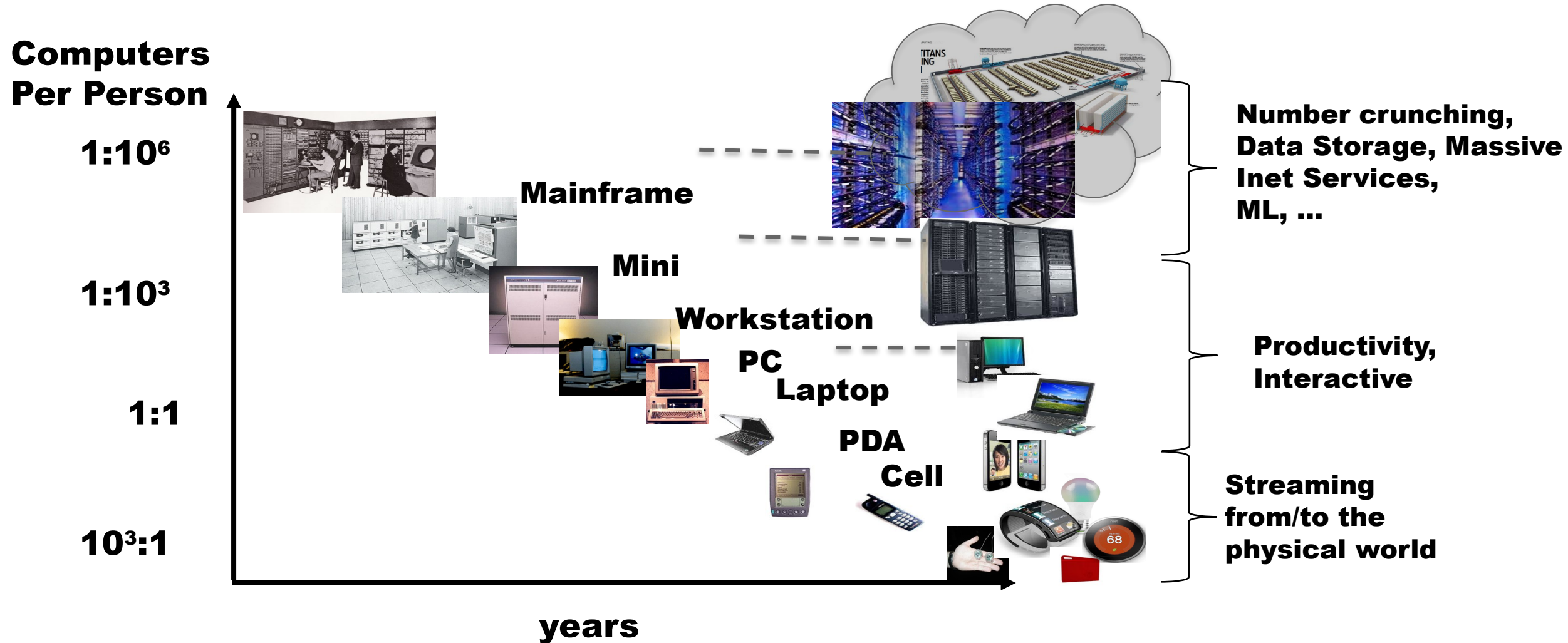
Have an operating system



Communicate over the Internet

Interface across huge diversity of devices

Bell's Law



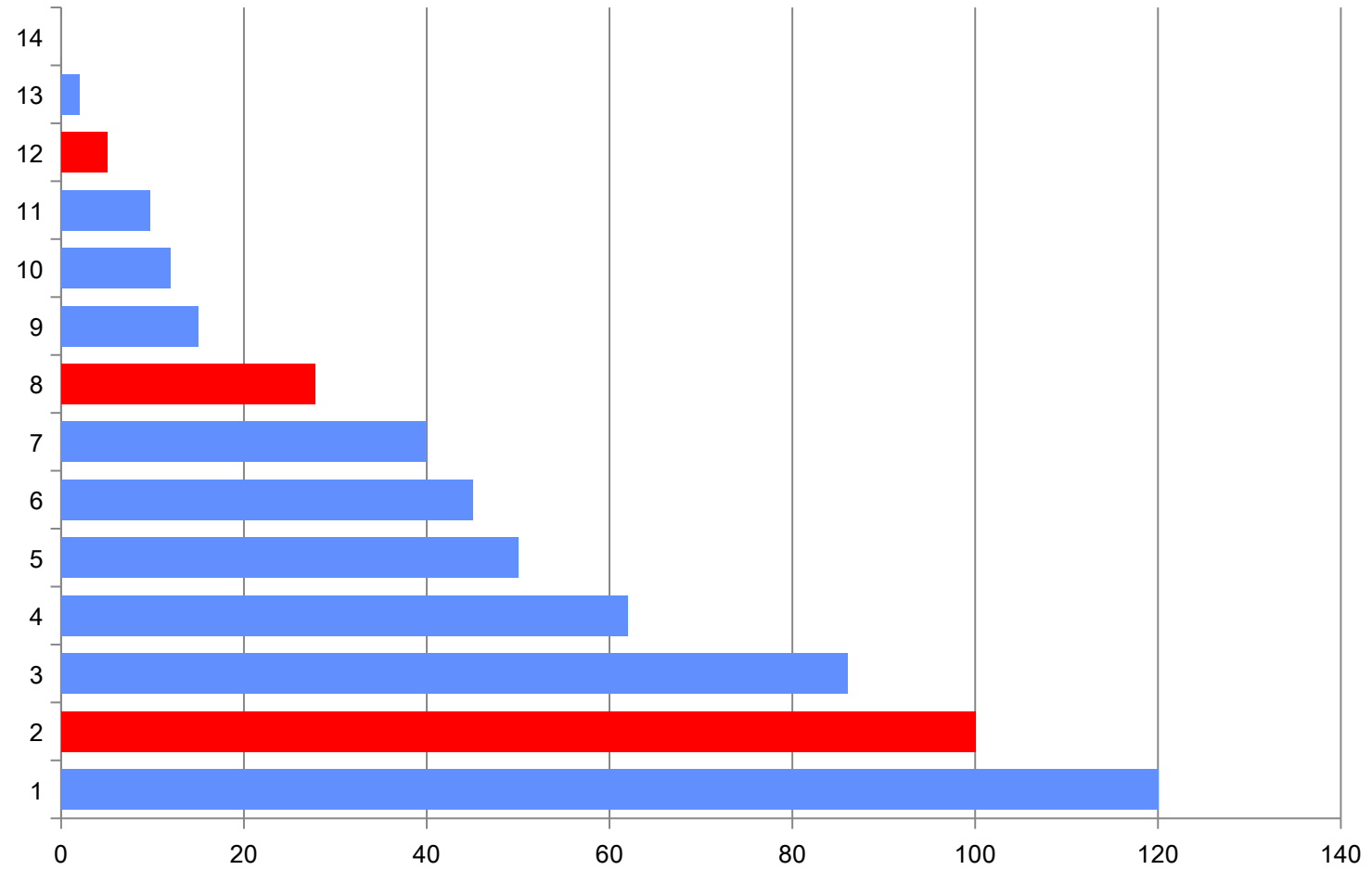
One new device class every 10 years

Across many timescales

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Jeff Dean's Numbers Everyone Should Know

With increased complexity



Why so much complexity?

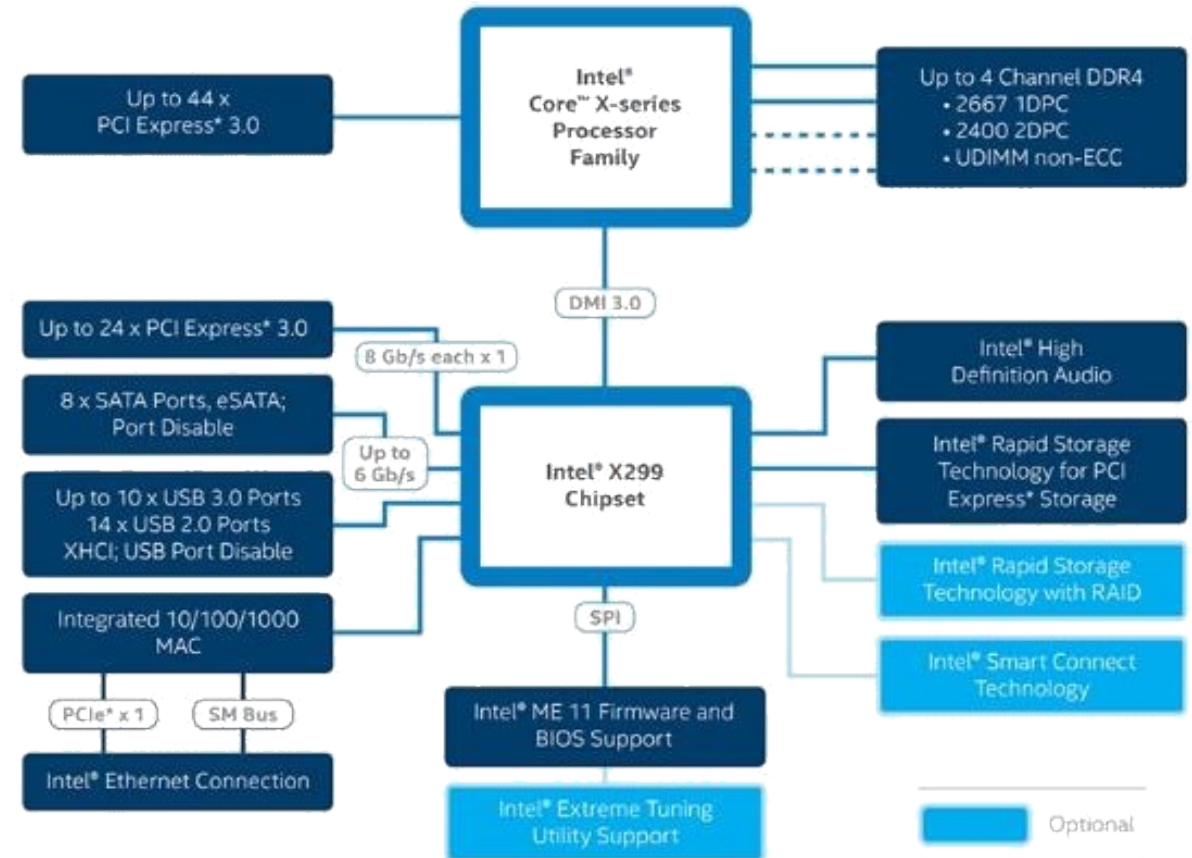
Hardware is becoming smarter!

Better reliability and security

Better performance (more efficient code, more parallel code)

Better energy efficiency

Legacy



Goal 3: What is an Operating System?

Operating System

Operating

Manages multiple tasks and users



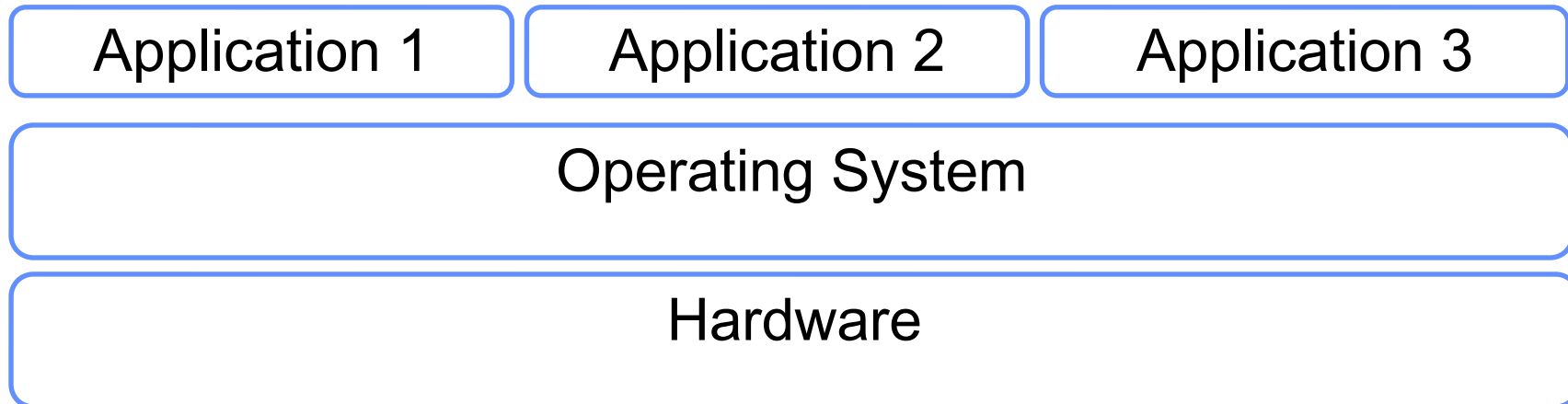
System

A set of interconnected components with an expected behaviour observed at the interface with its environment



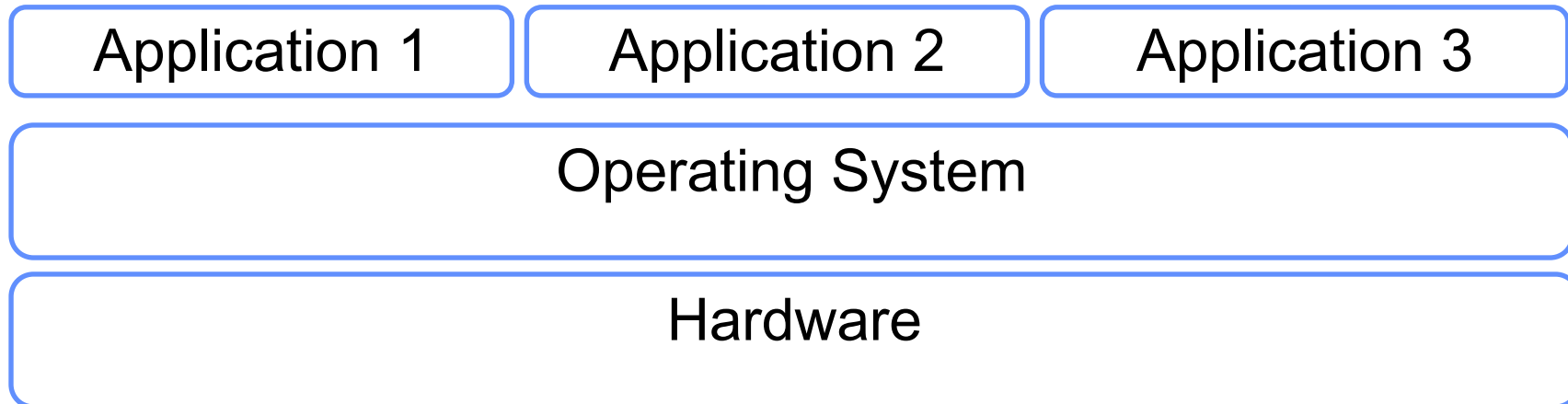
Operating System (v1)

An operating system is the layer of software that **interfaces** between (diverse) **hardware resources** and the (many) **applications** running on the machine



Operating System (v2)

An operating system implements a **virtual machine** for the application whose interface is more **convenient** than the raw hardware interface
(convenient = security, reliability, portability)



How I view an OS



Three main hats



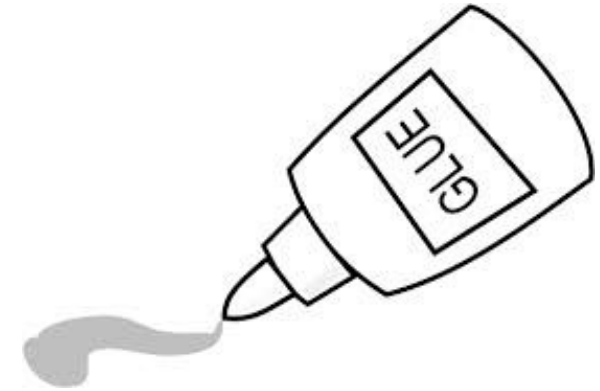
Referee

Manage protection, isolation, and sharing of resources



Illusionist

Provide clean, easy-to-use abstractions of physical resources

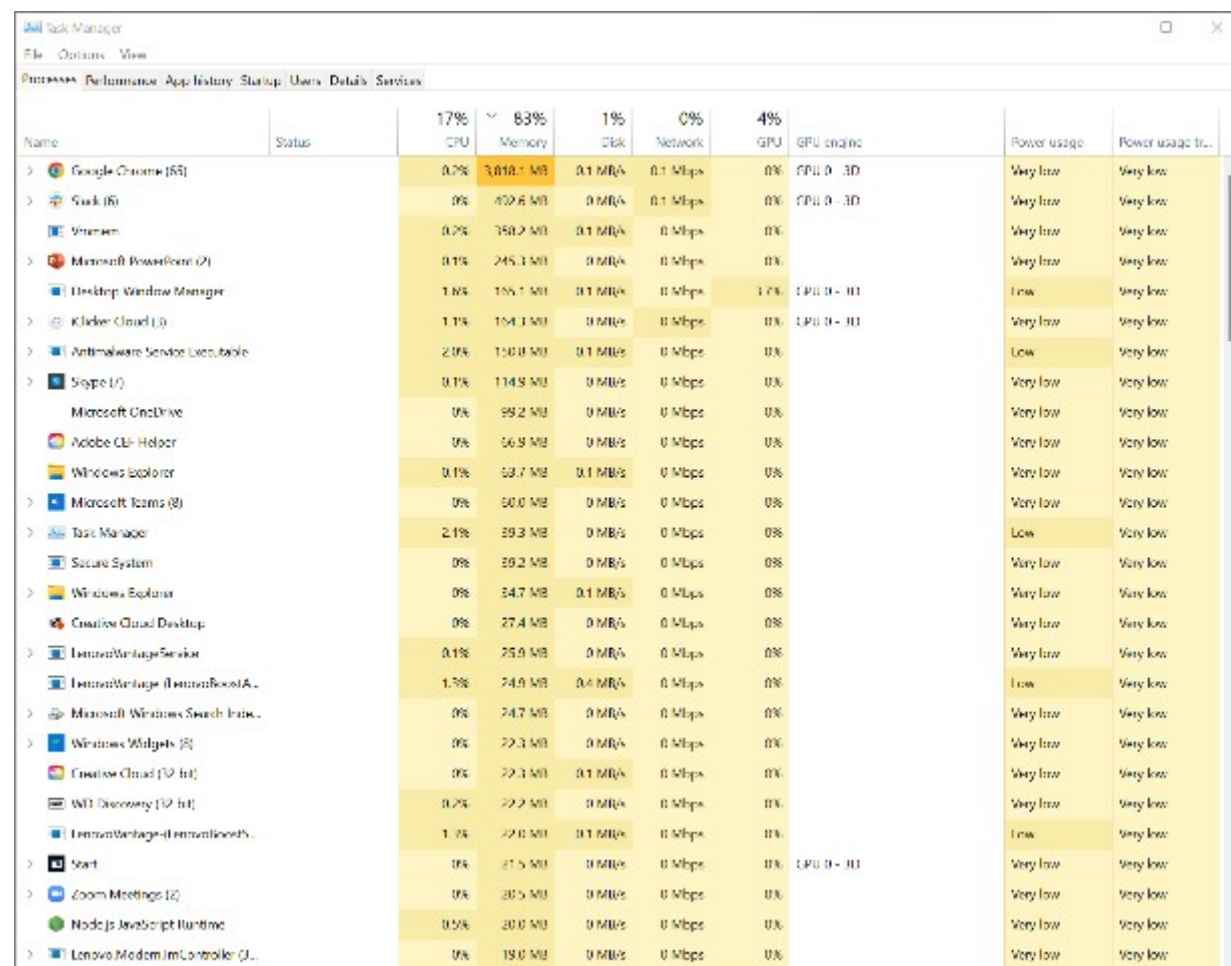


Glue

Provides a set of common services

OS as a referee

Allow multiple (untrusted) applications to run concurrently



The screenshot shows the Windows Task Manager Performance tab. The top bar displays overall system usage: CPU 17%, Memory 83%, Disk 1%, Network 0%, GPU 4%, and GPU engine. Below this is a table listing running applications and their resource usage.

Name	Status	CPU	Memory	Disk	Network	GPU	GPU engine	Power usage	Power usage fr...
Google Chrome [65]		0.2%	3,010.1 MB	0.1 MB/s	0.1 Mbps	0%	GPU 0 - 3D	Very low	Very low
Stack [6]		0%	402.6 MB	0 MB/s	0.1 Mbps	0%	GPU 0 - 3D	Very low	Very low
Visual Studio		0.2%	350.2 MB	0.1 MB/s	0 Mbps	0%		Very low	Very low
Microsoft PowerPoint [2]		0.1%	245.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Desktop Window Manager		1.6%	155.1 MB	0.1 MB/s	0 Mbps	1.7%	GPU 0 - 3D	Low	Very low
Clicker Cloud [3]		1.1%	154.1 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D	Very low	Very low
Antimalware Service Executable		2.0%	150.0 MB	0.1 MB/s	0 Mbps	0%		Low	Very low
Slingshot [2]		0.1%	114.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Microsoft OneDrive		0%	99.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Adobe CEF Helper		0%	66.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Explorer		0.1%	53.7 MB	0.1 MB/s	0 Mbps	0%		Very low	Very low
Microsoft Teams [8]		0%	50.0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Task Manager		2.1%	39.3 MB	0 MB/s	0 Mbps	0%		Low	Very low
Secure System		0%	39.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Explorer		0%	34.7 MB	0.1 MB/s	0 Mbps	0%		Very low	Very low
Creative Cloud Desktop		0%	27.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
LenovoVantageService		0.1%	25.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
LenovoVantage (LenovoFoodA...		1.5%	24.9 MB	0.4 MB/s	0 Mbps	0%		Low	Very low
Microsoft Windows Search Index...		0%	24.7 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Widgets [8]		0%	22.3 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Creative Cloud [12] Int		0%	22.3 MB	0.1 MB/s	0 Mbps	0%		Very low	Very low
WMI Discovery [12] Int		0.2%	22.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
LenovoVantage (LenovoFoodA...		1.7%	22.0 MB	0.1 MB/s	0 Mbps	0%		Low	Very low
Start		0%	21.5 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D	Very low	Very low
Zoom Meetings [2]		0%	20.5 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Node.js JavaScript Runtime		0.5%	20.0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Lenovo Modem (m) Controller [2]		0%	19.0 MB	0 MB/s	0 Mbps	0%		Very low	Very low

OS as a referee

Fault Isolation

Isolate programs from each other

Isolate OS from other programs

Process

Dual Mode Execution

Resource Sharing

How to choose which task to run next?

How to split physical resources?

Scheduling

Communication

How can OS support communication to share results?

Pipes/Sockets

What does this program do? (CS61C)

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <assert.h>

int main(int argc, char *argv[]){
    char *str = argv[1];
    while (1) {
        printf("%s\n", str);
    }
    return 0;
}
```

```
crooks@laptop> gcc -o cpu cpu.c -Wall
```

```
crooks@laptop> ./cpu A
```

A

A

A

A

...

```
crooks@laptop> ./cpu A & ./cpu B & ./cpu C
```

a)

A

A

A

A

...

b)

A

B

C

A

B

C

...

c)

A

B

B

A

C

B

C

...

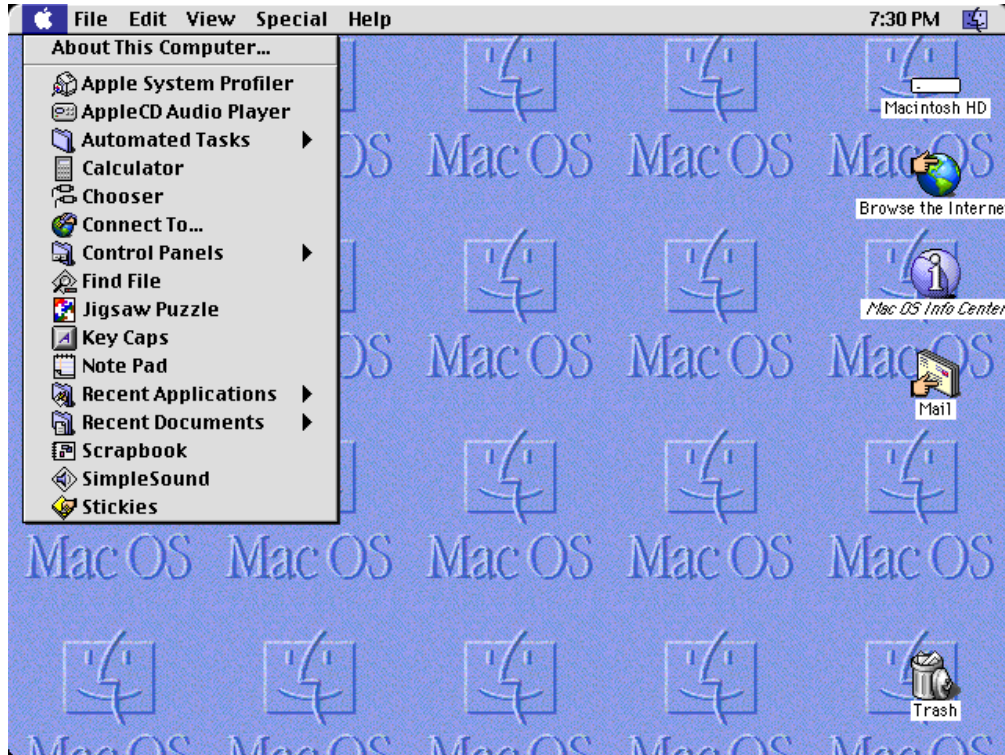
```
crooks@laptop> ./cpu & ; ./cpu B
```

Segmentation Fault

B

...

Refereeing is hard!



Mac V8 (1997)

OS cannot force program to give up control!

```
crooks@very-old-laptop>  
./cpu A & ./cpu B & ./cpu C
```

```
A  
A  
A  
A  
A  
A  
A  
...
```

Three main hats



Referee

Manage protection, isolation,
and sharing of resources



Illusionist

Provide clean, easy-to-
use abstractions of
physical resources

OS as Illusionist

Mask the restrictions inherent in computer hardware through [virtualization](#)

All alone

Provide abstraction that application has exclusive use of resources

All powerful

Provide abstraction that hardware resources are infinite

All expressive

Provide abstraction of hardware capabilities that are not physically present

What does this program do? (CS61C)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]){
    int *p = malloc(sizeof(int));
    printf("(%d) p: %p\n", getpid(), p);
    *p = 0;
    while (1) {
        *p = *p + 1;
        printf("(%d) p: %d\n", getpid(), *p);
    }
    return 0;
}
```

```
crooks@laptop> gcc -o memory memory.c -Wall
crooks@laptop> ./memory
```

```
(120) p: 0x200000
(120) p: 1
(120) p: 2
(120) p: 3
(120) p: 4
```

```
crooks@laptop> ./memory & ./memory
```

```
(120) p: 0x200000
(254) p: 0x200000
```

a)

```
(120) p: 1
(254) p: 2
(120) p: 3
(254) p: 4
(120) p: 5
(254) p: 6
```

...

b)

```
(120) p: 1
(254) p: 1
(120) p: 2
(254) p: 2
(120) p: 3
(254) p: 3
```

...

Three main hats



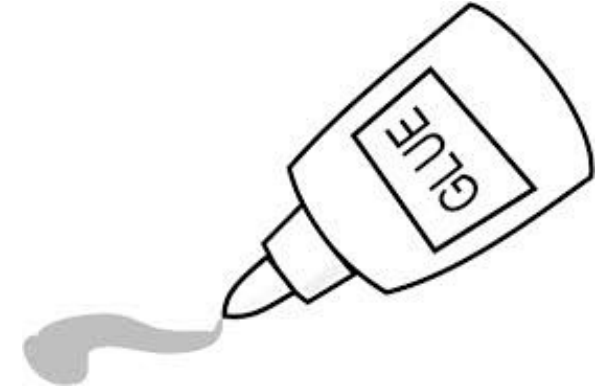
Referee

Manage protection, isolation, and sharing of resources



Illusionist

Provide clean, easy-to-use abstractions of physical resources



Glue

Provides a set of common services

OS as Glue

Provide set of common, standard services to applications to simplify and regularize their design

Make sharing easier

Simpler if all assume same
basic primitives

Maximise reuse

Avoid re-implementing
functionality from scratch.
Evolve components
independently

File System, User Interface, Network, etc.

Web browsers?

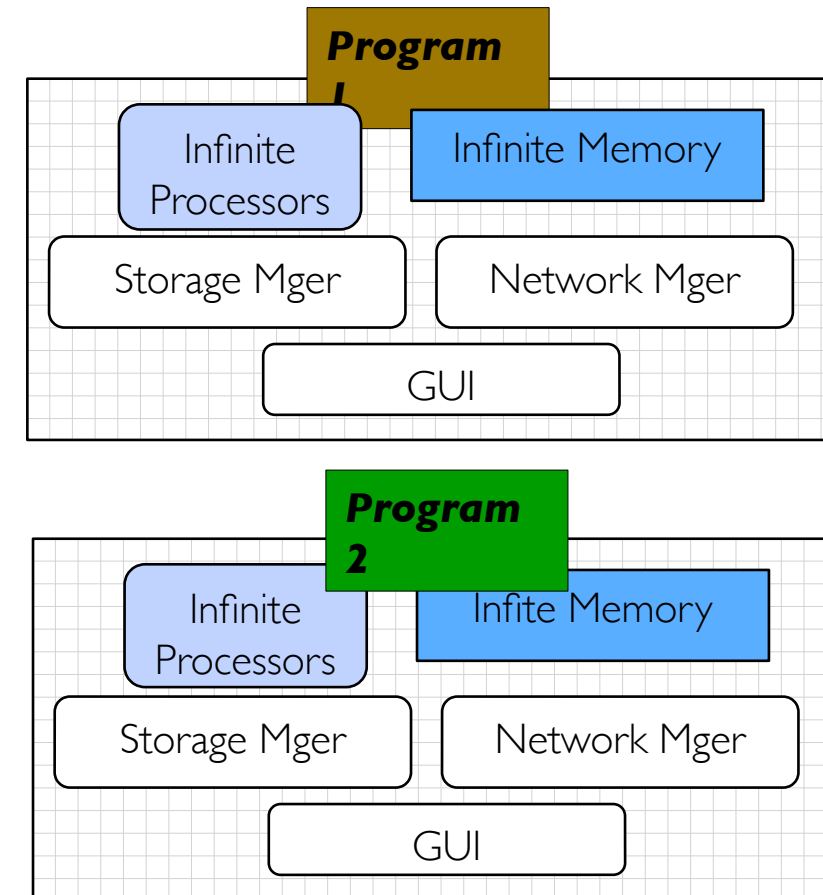
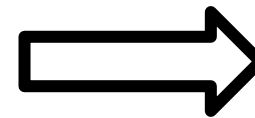
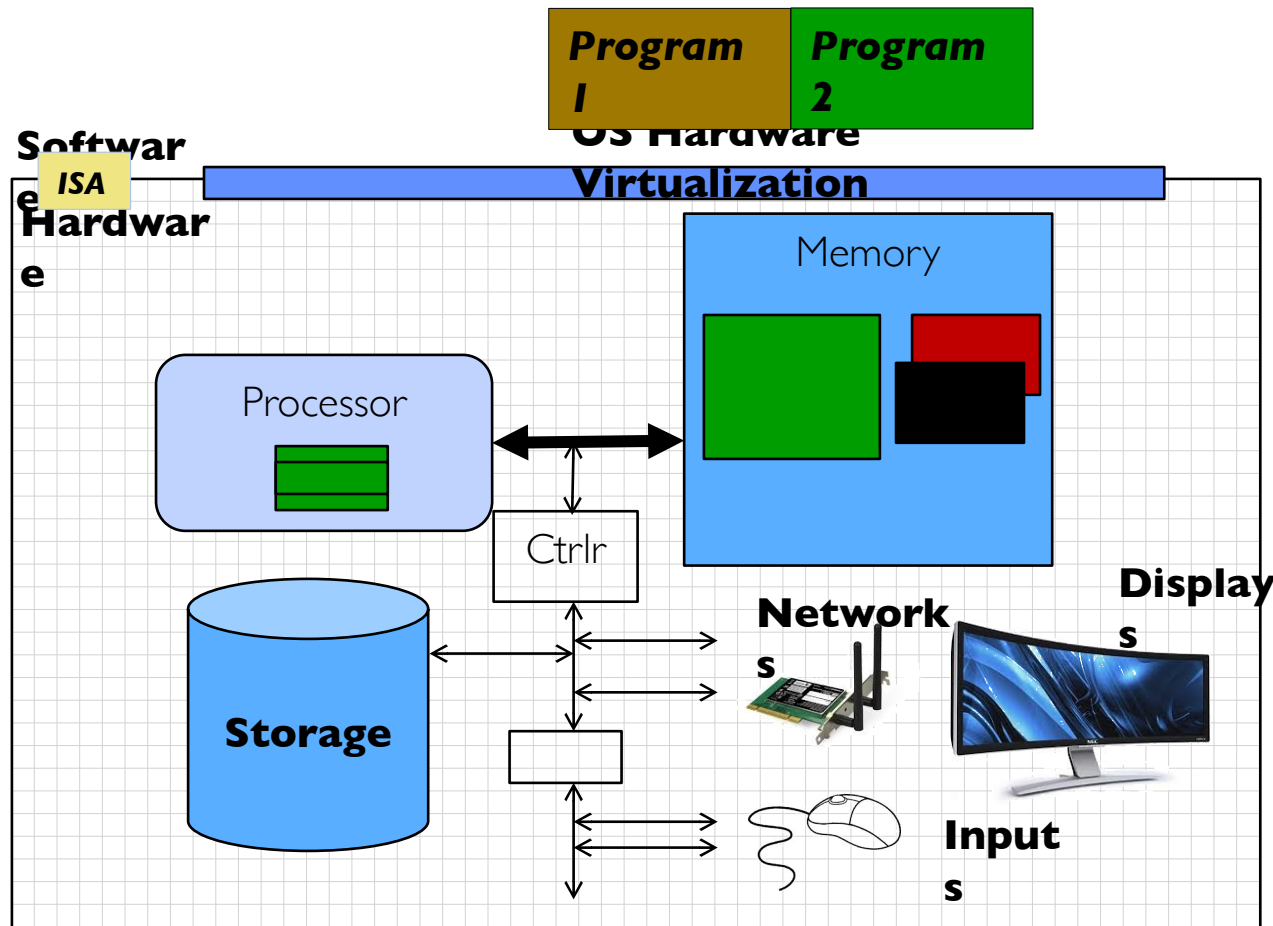
Are web browsers part of the OS?

United States of America v. Microsoft Corporation



Putting it all together

Referee + illusionist + Glue
=> Easy to use virtual machine



Evaluation Criteria: Performance

OS must implement the abstraction **efficiently**, with **low overhead**, and **equitably**

Overhead: added resource cost of implementing an abstraction

Fairness: How “well” are resources distributed across applications

Response time: how long does it take for a task to complete

Throughput: Rate at which group of tasks can be completed

Predictability: Are performance metrics constant over time?

Evaluation Criteria: Reliability

System does what it is supposed to do

OS failures catastrophic!



Availability: mean time to failure + mean time to repair

Evaluation Criteria: Security

Minimize vulnerability to attack

Integrity: Computer's operation cannot be compromised by a malicious attacker

Privacy: data stored on computer accessible to authorized users

Enforcement Policy

How the OS ensures only permitted actions are allowed

Security Policy

What is permitted



Evaluation Criteria: Portability

A portable abstraction **does not change** as the hardware changes

Can't rewrite application (or OS!) every time

Must plan for hardware that does not exist yet!

Application

Abstract Machine Interface

Operating System

Hardware Abstraction Layer

Hardware

Three “Prongs” for the Class

Understanding OS
principles

System Programming

Map Concepts to Real
Code

Topic Breakdown

Virtualizing the CPU

Process Abstraction and API

Threads and Concurrency

Scheduling

Virtualizing Memory

Virtual Memory

Paging

Persistence

IO devices

File Systems

Distributed Systems

Challenges with distribution

Data Processing & Storage

Enrollment

Class has 360 limit

- No guaranteed expansion

This is an Early Drop Deadline course (September 1st)

- If you are not serious about taking, please drop early
- Department will continue to admit students as other students drop
- Really hard to drop afterwards!
 - » Don't forget to keep up with work if you are still on the waitlist!

On the waitlist/Concurrent enrollment?

- If people drop, we can move others off waitlist
- Concurrent enrollment is after the waitlist

Infrastructure, Textbook & Readings

Infrastructure

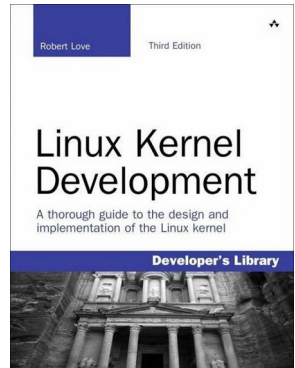
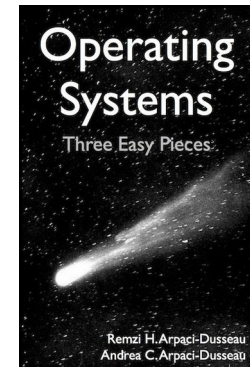
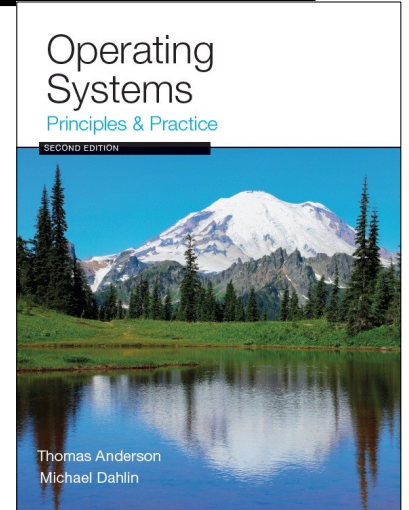
- Website: <http://cs162.org>
- EdStem: <https://edstem.org/us/courses/25794>

Textbook: Operating Systems: Principles and Practice (2nd Edition) Anderson and Dahlin

- Suggested readings posted along with lectures
- Try to keep up with material in book as well as lectures

Supplementary Material

- Operating Systems: Three Easy Pieces, by Remzi and Andrea Arpaci-Dusseau, available for free online
- Linux Kernel Development, 3rd edition, by Robert Love



Class Expectations

Lectures

- Come! Cannot guarantee content will be identical to previous years.
- Electronic devices used **only** for note taking.

Sections

- Attendance mandatory
- Meet your fellow students, they are your future colleagues!

Office Hours

- Come and ask for help early. No stupid question.
- We like teaching and want to meet you!

Communication Policy

Communicate with course staff through EdStem rather than Email.

Check Website for EdStem etiquette. Posts will be deleted if they do not follow guidelines

Learning by Doing

Individual Homeworks (2 weeks) - preliminary

0. Tools & Environment, Autograding, recall C, executable
1. Lists in C
2. BYOS – build your own shell
3. Memory & Management
4. Sockets & Threads in HTTP server
5. Distributed Systems

Three (and ½) Group Projects

0. Getting Started (Individual, before you have a group)
1. User-programs (exec & syscall)
2. Threads & Scheduling
3. File Systems



Group Projects

- Project teams have 4 members!
 - Never 5, 3 requires serious justification
 - Must work in groups in “the real world”
 - Same section (at least same TA)
- Everyone should do work and have clear responsibilities
 - You will evaluate your teammates at the end of each project
 - Dividing up by Task is the worst approach. Work as a team.
- Communicate with supervisor (TAs)
 - What is the team’s plan?
 - What is each member’s responsibility?
 - Short progress reports are required
 - Design Documents: High-level description for a manager!



Getting started

Start homework 0 and Project 0 right away when released

- Github account
- Registration survey
- Vagrant virtualbox – VM environment for the course
 - » Consistent, managed environment on your machine
- Get familiar with all the cs162 tools
- Submit to autograder via git

Sections on Wednesday – attend any section you want

- We'll assign permanent sections after forming project groups
- Section attendance will be mandatory after we form groups

Preparing Yourself for this Class

Projects will require you to be very comfortable with programming and debugging C

- Pointers (including function pointers, void*)
- Memory Management (malloc, free, stack vs heap)
- Debugging with GDB

You will be working on a larger, more sophisticated code base than anything you've likely seen in 61C!

Review Session on C/C++:

- TBA

"Resources" page on course website

- Ebooks on “git” and “C”

C programming reference (still in beta):

- <https://cs162.org/ladder/>
- First two sections are also dedicated to programming and debugging review. Attend ANY sections in first two weeks

Grading (Tentative breakdown)

36% three midterms (12% each)

36% projects

18% homework

10% participation (Sections, Lecture, ...)

Projects

- Initial design document, Design review, Code, Final design, Peer Review
- Submission via *git push* triggers autograder

Personal Integrity

UCB Academic Honor Code: "As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others."

<https://asuc.org/honor-code-landing/>

All academic violations will be reported

CS 162 Collaboration Policy



Explaining a concept to someone in another group
Discussing algorithms/testing strategies with other groups
Discussing debugging approaches with other groups
Searching online for generic algorithms (e.g., hash table)



Sharing code or test cases with another group
Copying OR reading another group's code or test cases
Copying OR reading online code or test cases from prior years
Helping someone in another group to debug their code

We compare all project submissions against prior year submissions and online solutions

Don't put a friend in a bad position by asking for help that they shouldn't give!

Course Environment

Implicit Bias

A form of bias that occurs automatically and unintentionally, that nevertheless affects judgments, decisions, and behaviors.

Every student has the right to learn and work in a safe environment

Ask yourself: I think/said X of/to student Y. Why?

Please reach out to **course staff** or **Title IX** office if there is an issue

Summary: Goals for Today

- Why should you care?
- Why is it hard?
- What is an Operating System?

Summary: Goals for Today

- Why should you care?

The OS is everywhere

- Why is it hard?

Deal with many different devices, many different time scales. Safety-critical

- What is an Operating System?

Provides abstraction of a simple, infinite virtual machine

Three roles: illusionist, referee and glue

A good OS cares about performance, reliability, security and portability