# CS 423
# Operating System Design: Introduction

Ram Alagappan

# Learning Objectives

**Before CS 423:**

• Knowledge of C/C++ • Basic knowledge of Linux/POSIX APIs and functions

**After CS 423:**

• Mastery of Operating Systems concepts • Comprehensive understanding of CPU and memory virtualization, concurrency problems and solutions, persistent storage • Become a kernel hacker capable of establishing a kernel development environment and modifying operating system code

**Today:**

• Introduce the instruction team • Go over the requirements and expectations

# Team

- Instructors

Prof. Ram Alagappan

- Teaching Assistants

Xuhao Luo (PhD student)

One more TA (hiring underway)

- Office Hours

Check the website

# Ram Alagappan

Call me “Ram”

Asst. Prof in CS (systems)

PhD from Wisconsin in 2019

Spent two years at VMware as a Researcher

Focus:

storage, distributed systems,

memory disaggregation

# TA: Xuhao Luo


• 4th Year Ph.D. Candidate
• Work with Prof. Ramnatthan & Prof. Aishwarya
• Research on distributed systems, storage systems, and other
• 2nd time TA CS423
• MS in Computer Science from UCSD in 2021
GES

# TA: 2nd TA

TBA

# Online Discussion Forum


**You are already added on the Piazza.**
**(if not, find the link on the course website)**

Go here for announcements and to ask questions.

Instruction team will be checking forums regularly!

# What’s in it for you?

Understand what is an operating system and why do we need one

Learn the internals of operating systems

Many concepts/ideas you learn here apply to systems software in general (not just OS)!

Core systems programming is becoming a rare skill in the industry

# Prereqs

Have you taken CS241?

Have you taken ECE391?

Do you have systems programming experience from another university or a job?

If not, you might find this course really hard…

# Textbook

“Operating Systems: Three Easy Pieces” OSTEP

Remzi and Andrea Arpaci-Dusseau

It is FREE! Available at ostep.org

The chapters are linked on the website.

# Other books

Alternative Textbooks (Not Free):

Operating Systems: Principles & Practice Anderson and Dahlin, 2018

Modern Operating Systems Tanenbaum and Bos, 2014

Operating System Concepts Silberschatz, Galvin and Gagne, 2012

Other Recommended Reading:

Linux Kernel Development Love, 2010 – Useful for MPs

# Requirements

Attendance/Participation

- Come to class, W/F, 2:00-3:15
- Participate actively in class and on piazza

Machine Problems (MPs): 4 major programming assignments

Midterm & Final Exams: Dates TBD

4 Credit Class: Read additional assigned literature and submit summaries weekly.

# Participation

Contribute in class — ask questions, respond to questions, share relevant outside knowledge.

Contribute *good* questions and answers on Piazza!

Other questions (e.g., administrative) on Piazza are also welcome, but won’t give you participation credit.

# Machine Problems

Implement and evaluate concepts from class in a commodity operating system

• Kernel Environment: Linux.

Not a toy OS, but a real 25 million LoC behemoth.

• Why? Building out a small OS is good experience, but navigating an existing code base is a more practical skill.

# Individual work

ALL WORK IS TO BE INDEPENDENTLY COMPLETED!

It’s okay to discuss MPs at a high level with others

Get help from TAs for MPs

It’s not okay to share code

It’s not okay directly help (with code or pair programming) others who haven’t finished yet

# MP Dev Environment

You will need a kernel development environment

MP 0: Setup a kernel development environment on your own machine

Linux machine?

Macbook?

Chromebook?

# MP Dev Environment

If you really don’t have a machine, we can ask Engr-IT to create a VM for you

But, historically, few did well if they rely on the VM

If you brick your machine (happens often), you will need to open a ticket with Engr-IT (>= 24 hour delay)

Brick your machine on a weekend? Too bad!

Occasionally, the VM cloud could just go down as well

# MP Submission

Code repository

You will need to submit your source code

We will create a private GitHub repo for you

Everything will be based on GitHub.

# 4 Cr Section

Intended audience: graduate students, ambitious undergraduate students interested in research

Earn your 4th credit by reading and summarizing weekly literature assignments Summaries due on the beginning of each class.

We will set up a google form or folder to submit your reviews

Assigned readings are marked as C4 in the class schedule

Grading: Summaries will contribute to C4 student's homework and participation credit.

# C4 Paper Summaries

Each summary should be about 1-2 pages in length.

• Structure your summary to cover:

1. What are the motivations for this work?
2. What is the proposed solution?
3. What is the work's evaluation of the proposed solution?
4. What is your analysis of the problem, idea and evaluation?
5. What are the contributions?
6. What are future directions for this research?
7. What questions are you left with?
8. What is your take-away message from this paper?

# Grading

Final Exam: 20% (format TBA)

Mid-term Exam: 20% (format TBA)

Machine Problems (5 total): 50% • 4%, 12%, 12%, 12%, 10%

Participation: 10%

Expect a few changes

# Policies

No late homework/MP submissions

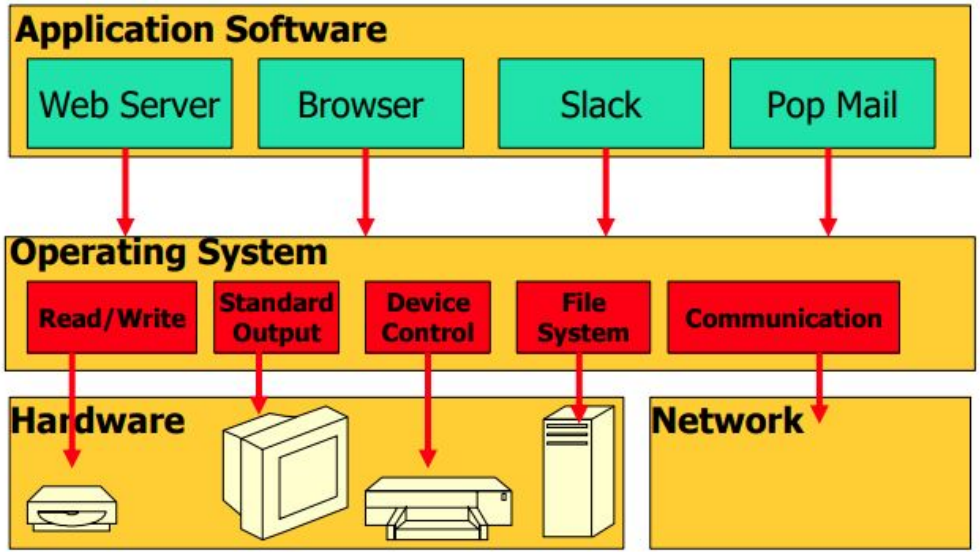
- 1 week window for re-grades from return date
- Cheating policy:

Zero tolerance

- 1st offense: get zero
- 2nd offense: fail class
- Example: You submitted two MPs in which solutions were not your own. Both were discovered at the same time. You fail class.

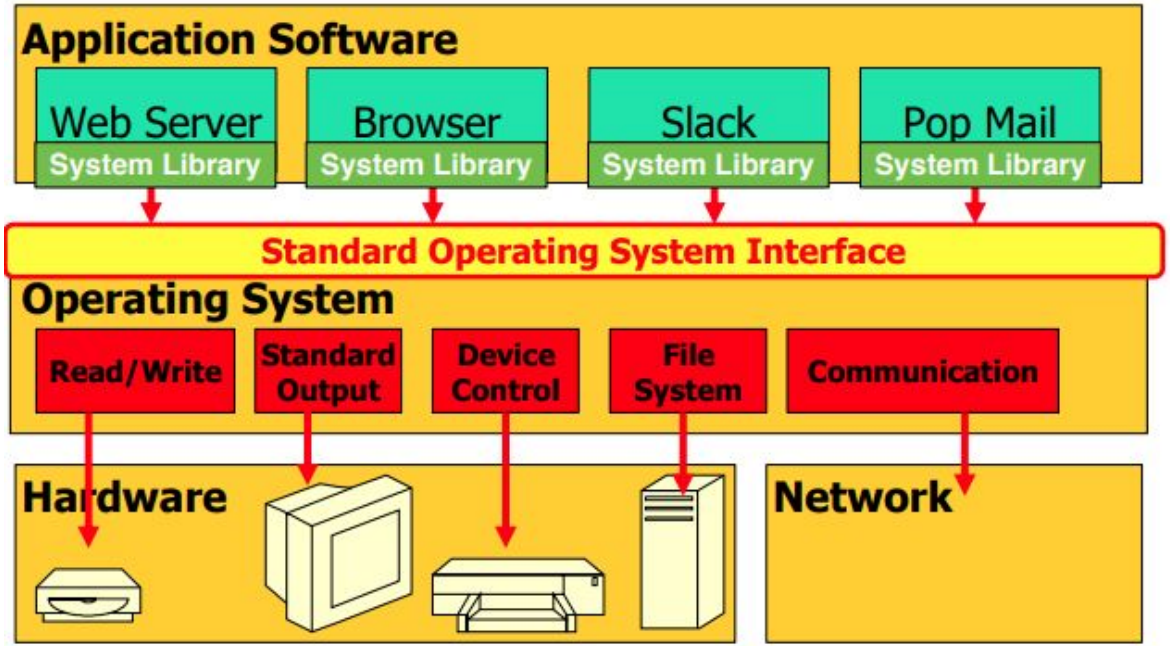
# What is an OS?

A layer of software that manages a computer's resources for its users and their applications

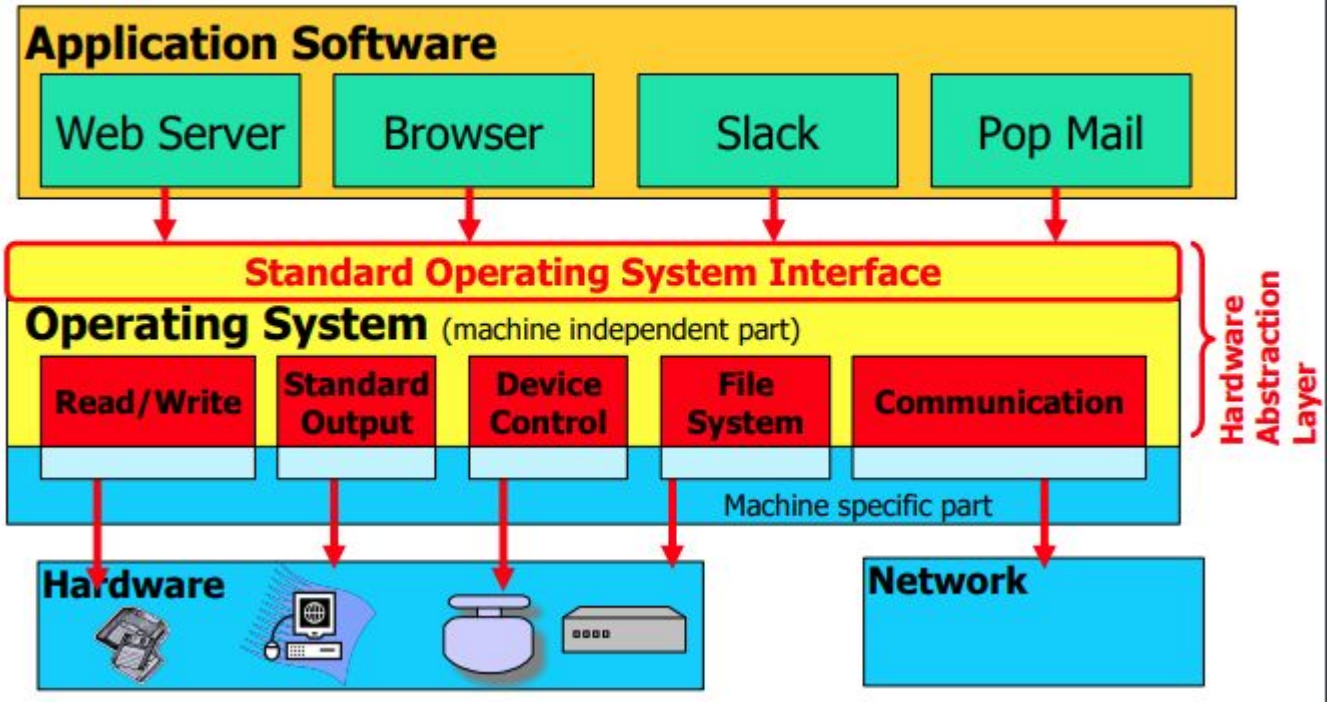


# OS Interface

***The OS exports a user interface. Why?***



OS Runs on Multiple Platforms while presenting the same Interface:



# WHAT DOES OS PROVIDE: ROLE #1

Abstraction: Provide standard library to access resources

What is a resource?

- Anything valuable (e.g., CPU, memory, disk)
- Examples of abstractions OS typically provide?
  - CPU:
  - Memory:
  - Disk:

# WHY SHOULD OS DO THIS ?

Advantages of OS providing abstraction?

- Allow applications to reuse common facilities
- Make different devices look the same
- Provide higher-level or more useful functionality

Challenges

- What are the correct abstractions?
- How much of hardware should be exposed?

# WHAT DOES OS PROVIDE: ROLE #2

Resource management – Share resources well

What is sharing?

- Multiple users of the system
- Multiple applications run by same user

# WHY SHOULD OS DO THIS ?

Advantages of OS providing resource management

- Protect applications at a common layer
- Provide efficient access to resources (cost, time, energy)
- Provide fair access to resources

Challenges

- What are the correct mechanisms?
- What are the correct policies?

# OPERATING SYSTEM ROLES SUMMARY

Two main roles

- Abstraction
- Resource management

Goals

- Ease of use
- Performance
- Isolation
- Reliability
- …

# COURSE APPROACH

# OPERATING SYSTEMS: THREE EASY PIECES

Three conceptual pieces

1. Virtualization

2. Concurrency

3. Persistence

# VIRTUALIZATION

Make each application believe it has each resource to itself

Example: CPU virtualization

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

What is the mechanism needed here?

What is the policy?

# VIRTUALIZATION

Make each application believe it has each resource to itself

Another Example: memory virtualization

# CONCURRENCY

Events occur simultaneously and may interact with one another

Need to

Provide abstractions (locks, semaphores, condition variables etc.)

# CONCURRENCY

```
static volatile int c = 0;
void *mythread(void *arg) {
int i;
for (i = 0; i < 1000000; i++) c++;
return NULL;
}
```

Main prints the value of c

What do you expect to be printed?

With 1 thread? With 2 threads?

What's going on here?

c++ boils down to something like this

```
mov mem_addr(c), eax
add 1, eax
mov eax, mem_addr(c)
```

Even on an uniprocessor!

# PERSISTENCE

Lifetime of data is longer than lifetime of any one process

Machine may lose power or crash unexpectedly

Issues:

- High-level abstractions: Files, directories (folders), links
- Correctness with unexpected failures
- Performance: disks are very slow!

# ADVANCED TOPICS

Virtualization

Concurrency

Persistence

Advanced Topics

- Virtual Machines
- Network File Systems
- SSDs

# Next Lecture

1/24 Friday

Topic: Process abstraction, CPU scheduling