CS 423
Operating System Design
Spring 2026

Ram Kesavan

Acknowledgement: U Wisconsin CS537

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

Staff

- Instructor

  Prof. Ram Kesavan

- Teaching Assistants

  Peizhe Liu (MS student)

  Gabriella Xue (PhD student)

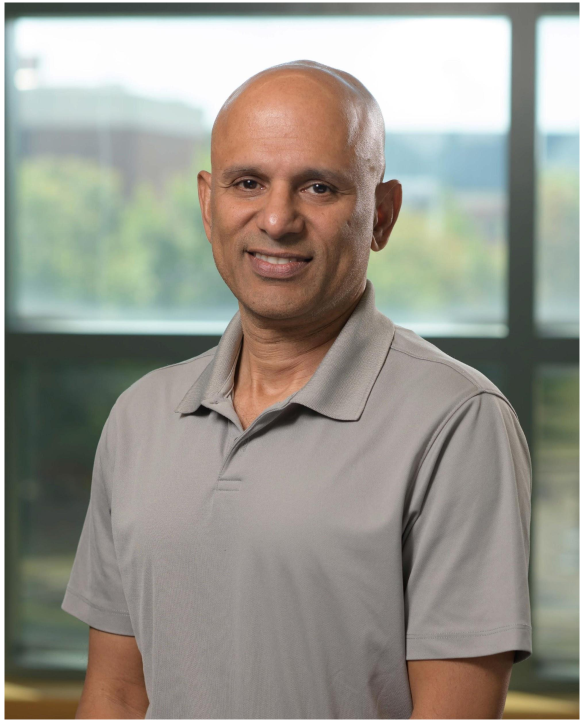
- Office Hours

  Check the website: will update soon

Ram Kesavan

Assoc. Clinical Prof

- 25 years in tech industry: NetApp & Google Cloud
- Storage, file systems, distributed databases
- Fall ‘25: started a new career in academia
- Interests: basketball, cycling, badminton, singing
- Still adjusting to the cold…brrr



Gabriella Xue

- PhD student working on satellite networking
- Favorite OS topic: networking
- Hobbies:
  - Marine life enthusiast
    - Interested in corals and saltwater fish
    - I keep 2 reef tanks with 40+ corals

Fun fact: if I can teach this little guy how to eat, then I'm not afraid of teaching anything :))

Peizhe Liu

2nd year M.S. student working on OS kernel and storage
Graduated from UIUC ECE
Hobbies
- Teaching
  (CS125, ECE385, ECE391, ECE411 as CA, CS423 as TA)
- Retrocomputing
  (Founder of Retrotech club at UIUC)
- Blood donation
  (Featured on local news)
- Homelab!

Online Discussion Forum

piazza

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!

Why take this course?

- Learn the internals of operating systems
- Most of the concepts/ideas apply to systems in general
- You’ll be faster & better at understanding, designing, and debugging most software & distributed systems
- Core systems becoming a rarer skill in the tech industry
- Necessary to call BS on boss’/AI’s/peers’ ideas

Prereqs

Have you taken CS341?

Have you taken ECE391?

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

If not, you may find this course very difficult…

Textbook

“Operating Systems: Three Easy Pieces” aka OSTEP
Remzi & 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 Cr: in a few slides

Participation

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

Contribute *good* questions and answers to Piazza!

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

Machine Problems

Implement and evaluate concepts learnt in a well-known OS

- 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 and debugging existing code is a more practical skill
  - Typical tech industry job: read & grok 100x to 1000x more code than you write
  - If anything, AI makes that ratio worse…it writes, you read!

Individual work

Design & Code TO BE DONE INDEPENDENTLY!

- Ok to discuss concepts & current Linux code with others & on Piazza
- Ok to discuss MPs at a high level with others & on Piazza
- Ok to get help from TAs for MPs; but they won’t design/debug your code
- Not ok to share code or design documents with

MP Dev Environment

All MPs need a Linux development environment
MP0: setup a kernel dev environment on your own machine

Linux or Windows or MacOS

We can also request VMs from EngrIT

Clearer instructions will come soon

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

Graduate & ambitious undergraduate students interested in research

Earn your 4th credit by reading and summarizing weekly literature assignments
Summaries due before start of Tue 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.

4Cr Paper Summaries

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

1. What were 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 were the contributions?
6. What were 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: 20-30%
- Mid-term: 20%
- Machine Problems: 50%--40%
    - Eg: 4%, 12%, 12%, 12%, 10%
- Participation: 10%

Might change a bit

Policies

No late MP submissions

- 1 week window for re-grades from return date

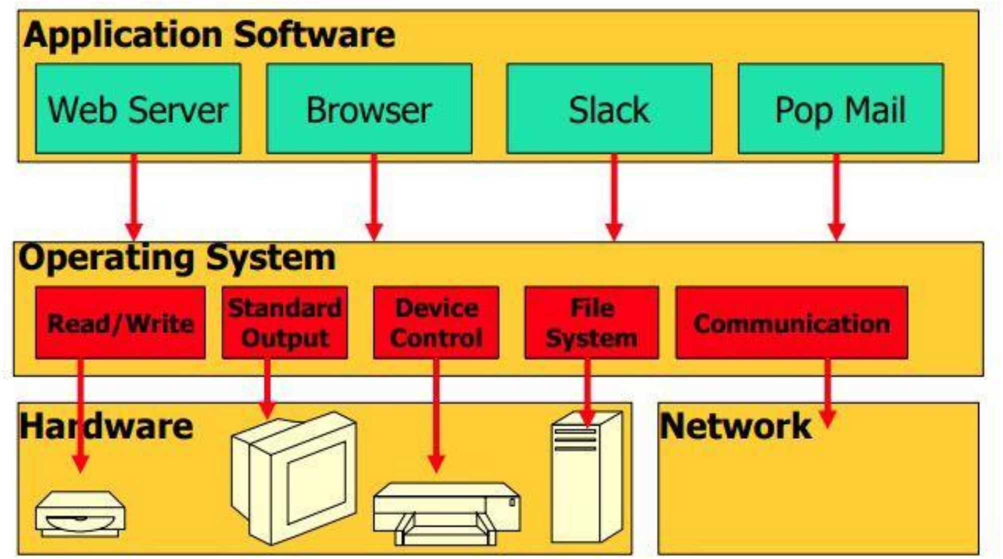
Cheating policy: Zero tolerance

- 1st offense: get zero
- 2nd offense: fail class

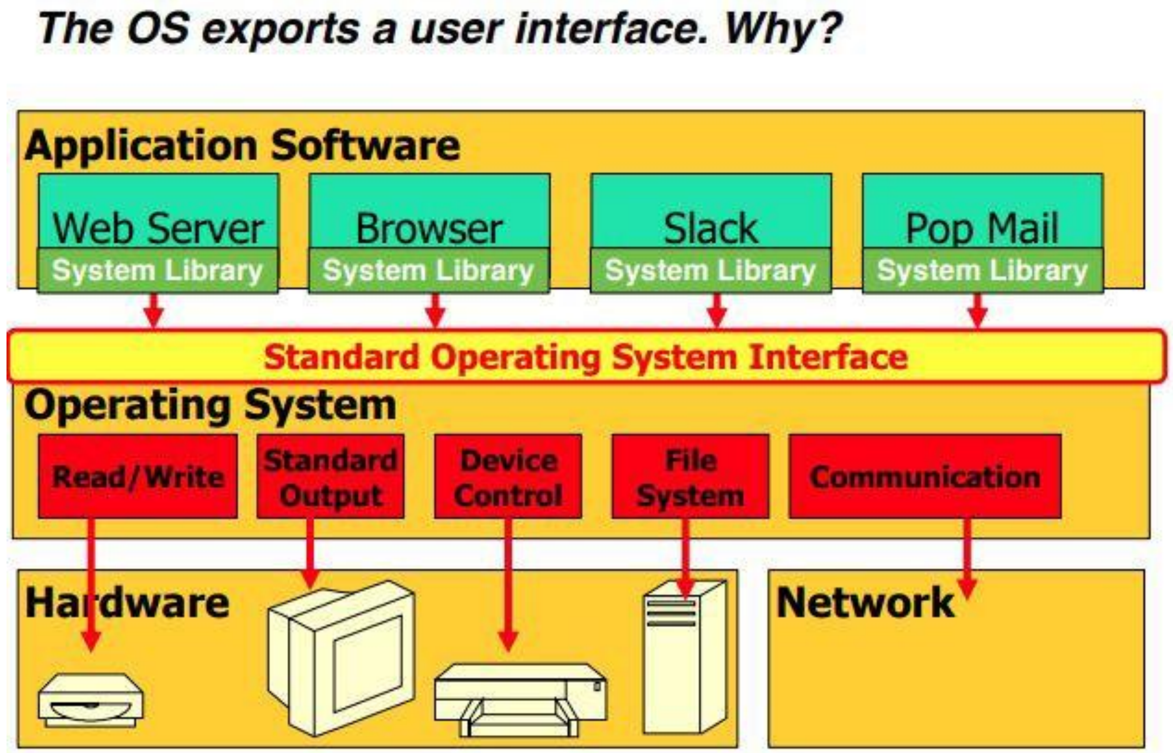
Example: You submitted two MPs in which solutions were not your own. Both were discovered at the same time. You fail class.

What is an OS?

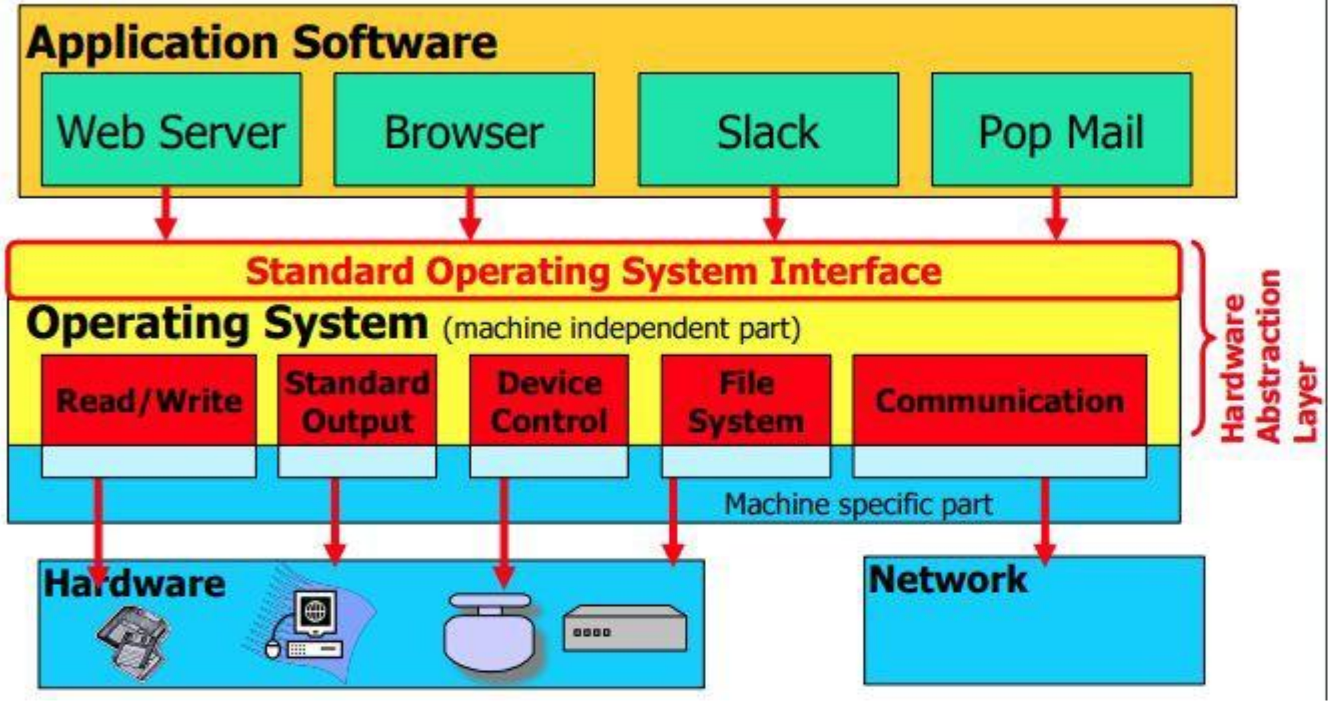
A layer of software that manages a computer’s resources for its users and their applications



OS Interface



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, storage)

Examples of abstractions OS typically provide?

CPU:

Memory:

Storage:

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

General-purpose OS: most concepts applicable to other kinds of OS

VIRTUALIZATION

Make each application believe it has each resource to itself

Example: CPU virtualization

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

What is the mechanism needed here?

What is the policy?

VIRTUALIZATION

Make each application believe it has each resource to itself

Virtualization also means isolation

Another Example: memory virtualization

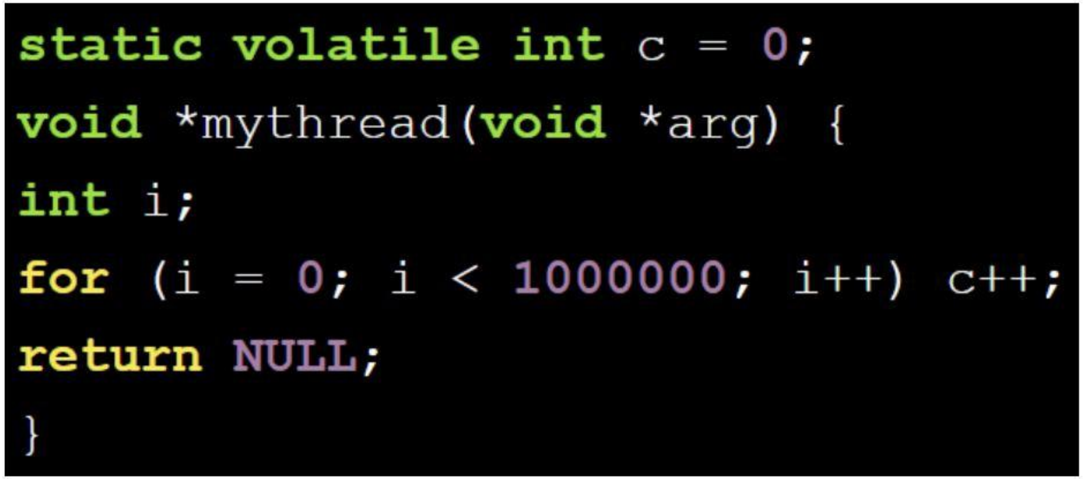
CONCURRENCY

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 happening here?

The line “c++;” when compiled produces:

```
// mov <dst>, <src>
mov eax, mem_addr(c)
add 1, eax
mov mem_addr(c), eax
```

What could go wrong?

PERSISTENCE

Data lives longer than execution lifetime of a one program

Machine may lose power or crash unexpectedly

Issues:

- High-level abstractions: Files, directories (folders), links
- Isolation: data ownership & sharing
- Correctness with unexpected failures
- Performance: disks are slow, SSDs faster

ADVANCED TOPICS

Virtualization

Concurrency

Persistence

Advanced Topics

- Virtual Machines
- Network File Systems
- SSDs

Today’s Class: Summary

Introduction to 423, staff, policies, etc.

General-purpose OS: what & why
3 pieces: virtualization, concurrency, persistence

Next Lecture

1/22 Thursday

Topic: Process abstraction, CPU scheduling