CS 423
Operating System Design

Tianyin Xu

* Thanks for Prof. Adam Bates for the slides.
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 virtualization techniques
• Introduction to advanced OS topics: security, energy, redundant 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 for this course
The Team

- Tianyin Xu (Instructor), tyxu@illinois.edu
- Siyuan Chai (TA), siyuanc3@illinois.edu
- Jinghao Jia (TA), jinghao7@illinois.edu
- Xuhao Luo (TA), xuhaol2@illinois.edu

Office Hour
- Mon 3:30--4:30pm Siebel 4108
- Tue 3:30--4:30pm Siebel 4108
- Wed 5:00-6:00pm Siebel 4108
- Thu 5:00--6:00pm Siebel 4108
I’m working on software and system reliability.

I worked at Facebook on dealing with datacenter level failures before joining UIUC.

- FAQ: academia or industry?
- I gained 20 LB eating free food.

I graduated from UC San Diego in 2017.

- I worked on hardening cloud and datacenter systems against misconfigurations.
- Dream job: a tenured grad student

I applied twice for grad school.

- I failed the first time (as always).
Siyuan Chai

- PhD student working on OS and memory systems
- Graduated from Northwestern in June 2021
- Hobbies
  - Soccer
    * Weekly on-campus soccer game for fun
    * Welcome to join!
  - Long distance running/cycling
  - Peking Opera
    * Try this video if you want to get a taste!
Jinghao Jia

- PhD student working on OS kernel extension & security
- Gentoo Linux user
- Hobby
  * Hacking & breaking stuff
  * Causing panics
  * “I’d rather be compiling.”
Xuhao Luo

- 2nd year PhD student
- Working with Prof. Tianyin Xu
- Research interests:
  * Distributed Systems
  * System reliability
- Education
  * MS from UCSD (Tianyin, Jinghao, and I are alumni!)
    - working on memory disaggregation
  * BS from USTC (China)
What’s in it for you?

• Understand the foundations of all system software

• Apply systems concepts and methodologies to higher layer software systems. Modern browsers, language virtual machines, and IoT devices all run their own forms of operating systems!

• Acquire a very particular (and lucrative) set of skills!

“I attended a Microsoft-organized meeting where the Director of Engineering of (Microsoft in Redmond) talked to me about a great need for engineers who know operating systems/device drivers, and know Linux kernel/programming at such lower levels. He bitterly complained that many CS departments are dismanteling their OS programs. I told him that we have actually multiple OS undergraduate classes at UIUC the current instructor to advertise among the students who take these courses that there are many jobs at Microsoft in OS area (more than ever!).”
• Prepare you for the real world!
  • Real-world software is imperfect and buggy
    • Unfortunately, you have to built on top of them;
  • Real-world infrastructure is fragile and vulnerable.
    • Unfortunately, you have to bear with that.
• Document is obsolete and even misleading
  • You won’t have teachers or TAs
• No matter what, have **FUN**!
Prerequisites

• Did you take CS241?
• Did you take ECE391?
• Do you have systems programming experiences from another university?
• If not, you might have a bad time in this course...
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!
“Operating Systems: Three Easy Pieces” OSTEP
Remzi and Andrea Arpaci-Dusseau

- It is **FREE**.
- **Why Textbooks Should Be Free**
- The chapters are linked on the website.
Additional Texts

• Alternative Textbooks (Not Free):
  Operating Systems: Principles & Practice
  \textit{Anderson and Dahlin}, 2018
  Modern Operating Systems
  \textit{Tanenbaum and Bos}, 2014
  Operating System Concepts
  \textit{Silberschatz, Galvin and Gagne}, 2012

• Other Recommended Reading:
  Virtual Machines
  \textit{Smith and Nair}, 2005
  Linux Kernel Development**
  \textit{Love}, 2010

** Helpful for MPs
CS 423 Requirements

- Attendance/Participation
  - Come to class, Tue/Thu, 2:00-3:15am
  - Participate actively in class and on piazza
- Machine Problems (MPs): 3 major programming assignments + 1 optional
- Periodic Homeworks: includes “prereqs” and “practice final”, may assign more
- Midterm & Final Exams: Dates TBD
- 4 Credit Class: Read additional assigned literature and submit summaries weekly.

ALL WORK IS TO BE INDEPENDENTLY COMPLETED!
Participation

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

• Contribute *good* questions and answers on Piazza!

• “The kind of answers you get to your technical questions depends as much on the way you ask the questions as on the difficulty of developing the answer.”

• **How To Ask Questions The Smart Way:**

• 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.

• Recall from earlier:

  “I attended a Microsoft-organized meeting where the Director of Engineering of (Microsoft in Redmond) talked to me about a great need for engineers who know operating systems/device drivers, and know linux kernel/programming at such lower levels. He bitterly complained that many CS departments are dismantling their OS programs. I told him that we have actually multiple OS undergraduate classes at UIUC the current instructor to advertise among the students who take these courses that there are many jobs at Microsoft in OS area (more than ever!).”
You will need a kernel development environment.

How many of you already setup the environment?

How many of you never worked in any OS stuff?

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

- Linux machine?
- Macbook?
- Chromebook?
• If you really don’t have a machine, we can ask Engr-IT to create a VM for you. But, historically, nobody did well if they are relying 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 for you.

• Occasionally, the VM cloud just goes down! That’s fun.
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.
Four Credit 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. The first summaries are due the first class of the 2\textsuperscript{nd} week (8/30).
- Print your summary out and bring it to the class.
- Assigned readings are marked as C4 in the Assignments section of the class schedule. Other students are not required to read these papers.
- 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?
Final Exam: 30%
Mid-term Exam: 25%
Homework: 4%
Machine Problems (4 total): 35%
  3%, 8%, 10%, 10%, 5%
Participation: 10%
  Class/Forum involvement
• **No late homework/MP submissions**
• 1 week window for re-grades from return date
• **Cheating policy: Zero tolerance**
  • 1\textsuperscript{st} offense: get zero
  • 2\textsuperscript{nd} 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.
Course Website

https://cs423-uiuc.github.io/fall22/

Go here for...

- Syllabus
- Course Schedule
- Lecture Slides/Recordings
- Links to other resources
What is an operating system?
Why Operating Systems?

Software to manage a computer’s resources for its users

Application Software
- Web Server
- Browser
- Slack
- Pop Mail

Operating System
- Read/Write
- Standard Output
- Device Control
- File System
- Communication

Hardware
- Network

CS423: Operating Systems Design
Why Operating Systems?

Software to manage a computer’s resources for its users

Application Software
- Web Server
  - System Library
- Browser
  - System Library
- Slack
  - System Library
- Pop Mail
  - System Library

Operating System
- Read/Write
- Standard Output
- Device Control
- File System
- Communication

Hardware
- Network
Why Operating Systems?

The OS exports a user interface. Why?

Application Software
- Web Server
- Browser
- Slack
- Pop Mail

Operating System
- Read/Write
- Standard Output
- Device Control
- File System
- Communication

Hardware

Network
Standard interface increases portability and reduces the need for machine-specific code.

Application Software

- Web Server
  System Library
- Browser
  System Library
- Slack
  System Library
- Pop Mail
  System Library

Standard Operating System Interface

Operating System (machine independent part)

- Read/Write
- Standard Output
- Device Control
- File System
- Communication

Machine specific part

Hardware

Network
OS Runs on Multiple Platforms while presenting the same Interface:

- **Application Software**
  - Web Server
  - Browser
  - Slack
  - Pop Mail

- **Operating System** (machine independent part)
  - Read/Write
  - Standard Output
  - Device Control
  - File System
  - Communication

- **Hardware**
- **Network**

Why Operating Systems?
OS Runs on Multiple Platforms while presenting the same Interface:
What are the responsibilities of an operating system?
Role #1: Referee

- Manage resource allocation between users and applications
- Isolate different users and applications from one another
- Facilitate and mediate communication between different users and applications
Role #2: Illusionist

- Allow each application to believe it has the entire machine to itself
- Create the appearance of an Infinite number of processors, (near) infinite memory
- Abstract away complexity of reliability, storage, network communication...
Role #3: Glue

- Manage hardware so applications can be machine-agnostic
- Provide a set of common services that facilitate sharing among applications

- Examples of “Glue” OS Services?
Role #3: Glue

- Manage hardware so applications can be machine-agnostic
- Provide a set of common services that facilitate sharing among applications

**Examples of “Glue” OS Services?**
- Cut-and-paste, File I/O, User Interfaces...
Consider file systems and storage devices…

How is the OS a referee? An illusionist? Glue?
Ex: File System Support

Referee
- Prevent users from accessing each other’s files without permission
- Even after a file is deleting and its space re-used

Illusionist
- Files can grow (nearly) arbitrarily large
- Files persist even when the machine crashes in the middle of a save

Glue
- Named directories, printf, other system calls for File I/O
What does an OS need to do in order safely run an untrustworthy application?
How should an operating system allocate processing time between competing uses?
Example: Web Service

- How does the server manage many simultaneous client requests?
- How do we keep the client safe from spyware embedded in scripts on a web site?
- How do handles updates to the web site such that clients always see a consistent view?
OS Challenges

Reliability
• Does the system do what it was designed to do?

Availability
• What portion of the time is the system working?
• Mean Time To Failure, Mean Time to Repair

Security
• Can the system be compromised by an attacker?

Privacy
• Data is accessible only to authorized users
Portability

- For programs:
  Application programming interface (API)
  Abstract virtual machine (AVM)
- For hardware
  Hardware abstraction layer
OS Challenges

Performance
  Latency/response time
    How long does an operation take to complete?
Throughput
  How many operations can be done per unit of time?
Overhead
  How much extra work is done by the OS?
Fairness
  How equal is the performance received by different users?
Predictability
  How consistent is the performance over time?
OS Family Tree

- **MS/DOS**
  - **Windows**
    - **Windows NT**
      - **Windows 8**
    - **VMware**
  - **VMS**
  - **VM/370**

- **MVS**

- **Multics**
  - **BSD UNIX**
    - **Linux**
    - **Android**
  - **Mach**
    - **NEXT**
    - **MacOS**
    - **MacOS X**

- **Descendant**
- **Influence**
## Performance / Time

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniprocessor speed (MIPS)</td>
<td>1</td>
<td>200</td>
<td>2500</td>
<td>2.5K</td>
</tr>
<tr>
<td>CPUs per computer</td>
<td>1</td>
<td>1</td>
<td>10+</td>
<td>10+</td>
</tr>
<tr>
<td>Processor MIPS/$</td>
<td>$100K</td>
<td>$25</td>
<td>$0.20</td>
<td>500K</td>
</tr>
<tr>
<td>DRAM Capacity (MiB)/$</td>
<td>0.002</td>
<td>2</td>
<td>1K</td>
<td>500K</td>
</tr>
<tr>
<td>Disk Capacity (GiB)/$</td>
<td>0.003</td>
<td>7</td>
<td>25K</td>
<td>10M</td>
</tr>
<tr>
<td>Home Internet</td>
<td>300 bps</td>
<td>256 Kbps</td>
<td>20 Mbps</td>
<td>100K</td>
</tr>
<tr>
<td>Machine room network</td>
<td>10 Mbps (shared)</td>
<td>100 Mbps (switched)</td>
<td>10 Gbps (switched)</td>
<td>1000</td>
</tr>
<tr>
<td>Ratio of users to computers</td>
<td>100:1</td>
<td>1:1</td>
<td>1:several</td>
<td>100+</td>
</tr>
</tbody>
</table>
One application at a time

- Had complete control of hardware
- OS was runtime library
- Users would stand in line to use the computer

Batch systems

- Keep CPU busy by having a queue of jobs
- OS would load next job while current one runs
- Users would submit jobs, and wait, and wait, and wait,
Multiple users on computer at same time

- Multiprogramming: run multiple programs at same time
- Interactive performance: try to complete everyone’s tasks quickly
- As computers became cheaper, more important to optimize for user time, not computer time
Today’s OSs

- Smartphones
- Embedded systems
- Laptops
- Tablets
- Virtual machines
- Data center servers
Tomorrow’s OSs

• Giant-scale data centers
• Increasing numbers of processors per computer
• Increasing numbers of computers per user
• Very large scale storage
**Your To-Do List**

**Today:**
- Visit the class webpage and check out all the info
  - [https://cs423-uiuc.github.io/spring20/](https://cs423-uiuc.github.io/spring20/)
  - Refresh your system programming skills (e.g., review CS 241 and see C language tutorial below)
  - [http://www.lysator.liu.se/c/bwk-tutor.html](http://www.lysator.liu.se/c/bwk-tutor.html)
- Familiarize yourself with Piazza

**Soon:**
- Access CS 423 development VM, begin MP0
- Complete HW0