CS 423
Operating System Design: Overview and Basic Concepts

Professor Adam Bates
Fall 2018
Goals for Today

• **Learning Objectives:**
  • Introduce OS definition, challenges, and history

• **Announcements:**
  • C4 readings for Week 2 are out! **Due Jan 25 (UTC-11)**
  • HW0 is available on Compass! **Due Jan 25 (UTC-11)**
  • MP0 is available for review on Compass! **Due Jan 27 (UTC-11)**

**Reminder:** Please put away devices at the start of class
Goals for Today

• **Announcements continued:**
  • TA Office Hours: Monday 3-5pm
    • Room: Siebel Center 0207
    • *Go here for MP questions!*

*Reminder:* Please put away devices at the start of class

Alberto (TA)
Piazza Questions

Can I register for the C4 class?

HW0 not available!
What is an operating system?
Why Operating Systems?

Software to manage a computer’s resources for its users and applications.

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

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

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

**Network**
Why Operating Systems?

Software to manage a computer’s resources for its users and applications.

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

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

### Hardware
- Network

---

CS 423: Operating Systems Design
The OS exports a user interface. Why?

Why Operating Systems?

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

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

Hardware
- Printers

Network
Standard interface increases portability and reduces the need for machine-specific code.
Why Operating Systems?

OS Runs on Multiple Platforms while presenting the same Interface:

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

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

Operating System (machine independent part)

Hardware Abstraction Layer
- Machine specific part

Hardware

Network
What are the responsibilities of an operating system?
Operating System Roles

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 (MTTF), 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

![Diagram showing the relationship between users, kernel-mode, user-mode, system library, and system components like file system, virtual memory, networking, scheduling, and hardware-related components like processors, address translation, and network.]
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

- MVS
  - MS/DOS
    - Windows
      - Windows NT
        - Windows 8
  - VMS
  - VM/370

- Multics
  - UNIX
    - BSD UNIX
      - VM/370
    - Mach

- UNIX
  - Linux
    - Android
  - NEXT
    - MacOS
  - MacOS X
    - iOS
## 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>

Figure 1.8: Approximate computer server performance over time, reflecting the most widely used servers of each era: in 1981, a minicomputer; in 1997, a high-end workstation; in 2014, a rack-mounted multicore server. MIPS stands for "millions of instructions per second," a measure of processor performance. The VAX 11/782 was introduced in 1982; it achieved 1 MIP. DRAM prices are from Hennessey and Patterson, "Computer Architecture: A Quantitative Approach." Disk drive prices are from John McCallum. The Hayes smartmodem, introduced in 1981, ran at 300bps. The 10 Mbps shared Ethernet standard was also introduced in 1981. One of the authors built his first operating system in 1982, used a VAX at his first job, and owned a Hayes to work from home.

Despite these changes, operating systems still face the same conceptual challenges as they did fifty years ago. To manage computer resources for applications and users, they must allocate resources among applications, provide fault isolation and communication services, abstract hardware limitations, and so forth. Tremendous progress has been made towards improving the reliability, security, efficiency, and portability of operating systems, but much more is needed. Although we do not know precisely how computing technology or application demand will evolve over the next 10-20 years, it is highly likely that these fundamental operating system challenges will persist.

Early Operating Systems

Computers were expensive; users would wait. The first operating systems were runtime libraries intended to simplify the programming of early computer systems. Rather than the tiny, inexpensive yet massively complex hardware and software systems of today, the first computers often took up an entire floor of a warehouse, cost millions of
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