How This Course Fits in the UW CSE Curriculum

• CSE 333: Systems Programming
  – Project experience in C/C++
  – How to use the operating system interface

• CSE 451: Operating Systems
  – How to make a single computer work reliably
  – How an operating system works internally

• CSE 452: Distributed Systems
  – How to make a set of computers work reliably, despite failures of some nodes
Project: OS/161

• Build an operating system
  – That can boot on a multiprocessor
• We give you some basic building blocks
  – Three assignments, that build on each other
    • Threads, user programs, virtual memory
  – Work in groups of 2-3 (recommend 3!)
• Instructions on web page
  – Download and browse code before section
  – Bring laptop or smartphone to section
• Assignment 0 due next Thursday
Problem Sets

• Two parts, equivalent to a final
  – Done individually
Main Points (for today)

• Operating system definition
  – Software to manage a computer’s resources for its users and applications

• OS challenges
  – Reliability, security, responsiveness, portability, ...

• OS history
  – How are OS X, iOS, Windows 10, Android, and Linux related?
What is an operating system?

- Software to manage a computer’s resources for its users and applications
Operating System Roles

• Referee:
  – Resource allocation among users, applications
  – Isolation of different users, applications from each other
  – Communication between users, applications

• Illusionist
  – Each application appears to have the entire machine to itself
  – Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport

• Glue
  – Libraries, user interface widgets, ...
Example: File Systems

• Referee
  – Prevent users from accessing each other’s files without permission
  – Even after a file is deleted 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, ...
Question

• What (hardware, software) do you need to be able to run an untrustworthy application?
Question

• How should an operating system allocate processing time between competing uses?
  – Give the CPU to the first to arrive?
  – To the one that needs the least resources to complete? To the one that needs the most resources?
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 make updates to the web site so 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
OS Challenges

- **Portability**
  - For programs:
    - Application programming interface (API)
    - Abstract virtual machine (AVM)
  - For the operating system
    - 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 History

- **MS/DOS**
  - Windows
    - Windows NT
      - Windows 8
    - VMWare
- **VMS**
- **VM/370**
- **Multics**
- **UNIX**
  - BSD UNIX
  - Mach
  - Linux
    - Android
  - NEXT
  - MacOS
- **MVS**
- **UNIX**

- **Influence**
- **Descendant**
Computer Performance Over 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 300 bps. 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.
Early Operating Systems: Computers Very Expensive

• 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
Time-Sharing Operating Systems: Computers and People Expensive

• 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 Operating Systems: Computers Cheap

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

- Giant-scale data centers
- Increasing numbers of processors per computer
- Increasing numbers of computers per user
- Very large scale storage
Textbook

• Lazowska, Spring 2012: “The text is quite sophisticated. You won't get it all on the first pass. The right approach is to [read each chapter before class and] re-read each chapter once we've covered the corresponding material... more of it will make sense then. Don't save this re-reading until right before the mid-term or final – keep up.”