CSE 451 15sp
Operating Systems

John Zahorjan

Today's Outline

- Course administration
  - Staff
  - Communication
  - Workload
- Introduction to OS
  - Chapter 1

Course Staff

- Sean Flinn

Sean Flinn

- Bran Hagger

- Kendall Lowrey

Kendall Lowrey

Course Communication

- Staff -> you
  - Email (!)
    - *cse451a_sp15@uw.edu*
  - goPost forum
- Staff <-> you
  - In class, in office hours, in the hall
  - goPost, email
- You <-> you
  - ?

Course Workload

- Lectures, sections
- Reading
- Homeworks
- Projects
  - 0: code reading
  - 1: Sync primitives (lock, cond var, thread join); thread safe data structures
  - 2: processes; multitasking
  - 3: virtual memory; paging

Late Policy

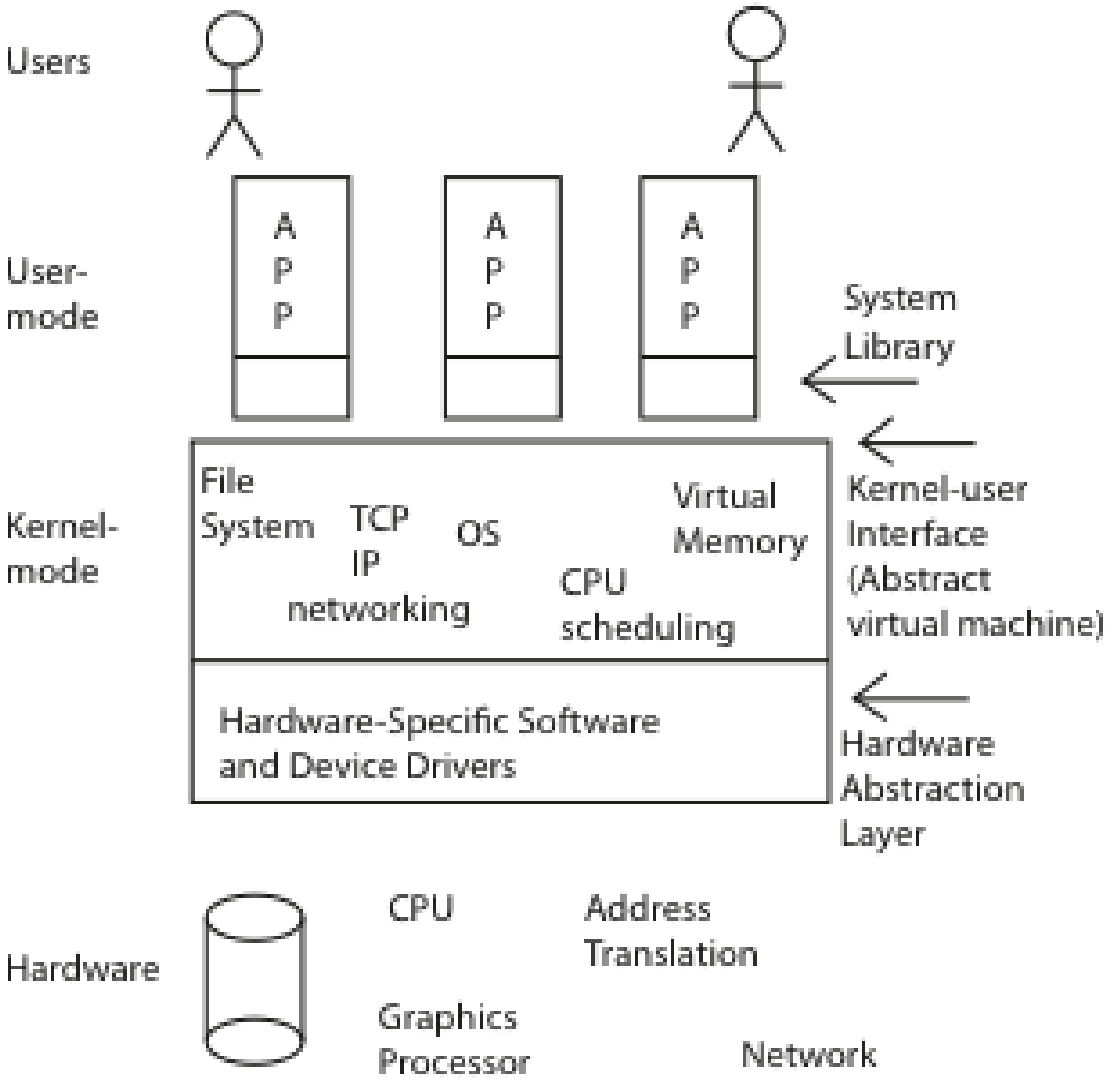
- 4 slip days
- Hard deadline at end of quarter
- Hack days…

Operatin Systems: 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, Windows 7, and Linux related?

What is an operating system?

- Software to manage a computer's resources for its users and applications



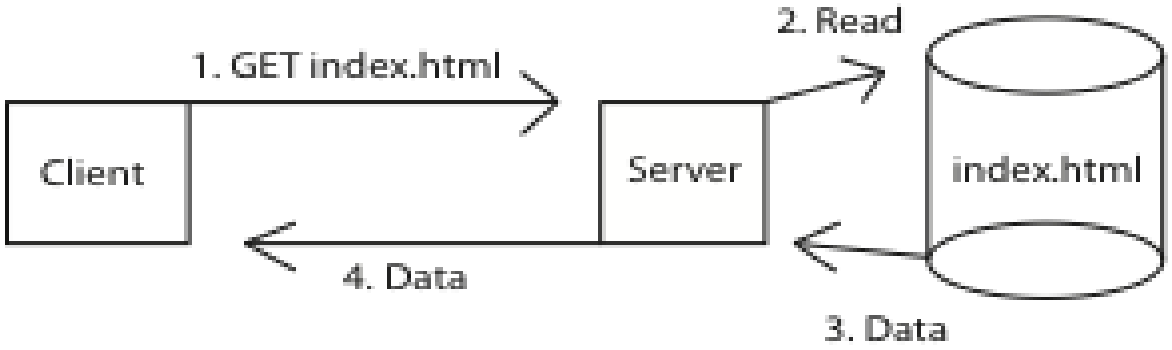
Operating System Roles

- 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 file-based storage, reliable network transport
- Glue
  - Libraries, user interface widgets, …

Thought Question

- What do you need from hardware to be able to:
  - Isolate different applications from each other?
  - Isolate different users from accessing each others files?

Example: web service



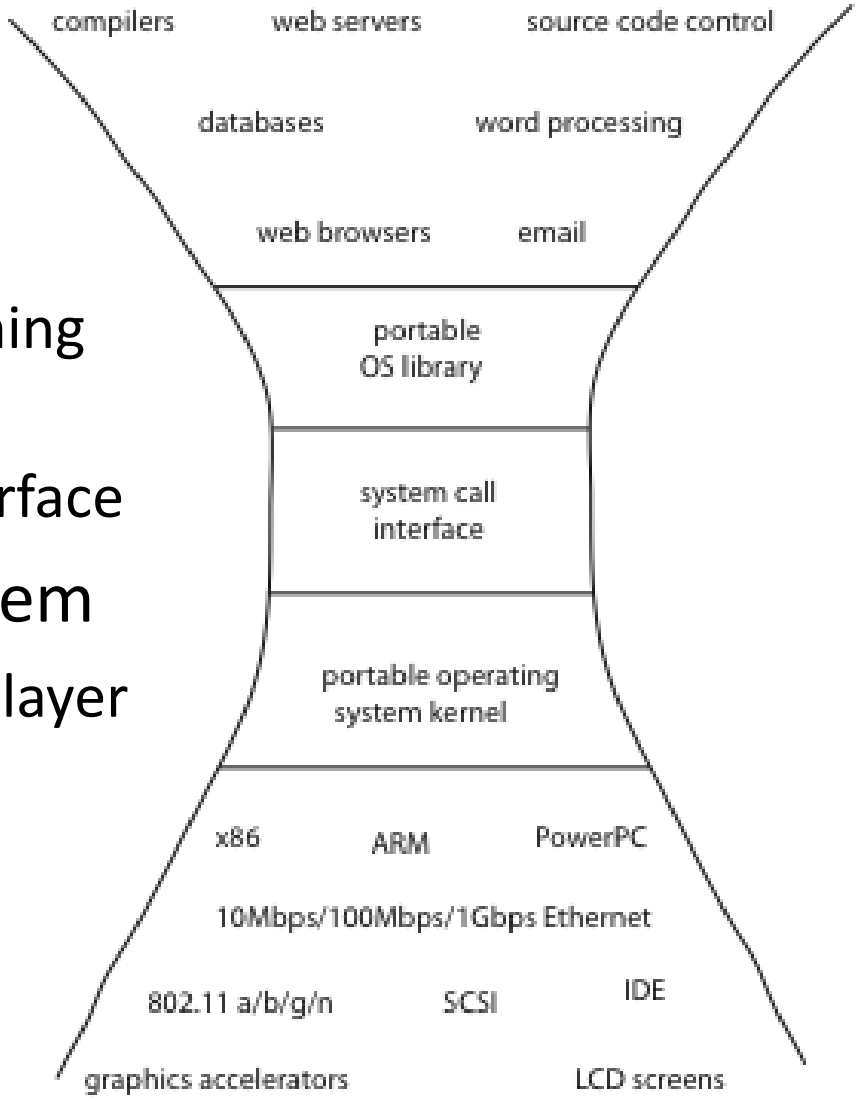
- How does the server manage many simultaneous client requests?
- How do we keep the client safe from spyware embedded in scripts on a web site?
- How do we keep updates to the web site consistent?

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
- *Both require very careful design and code*

OS Challenges

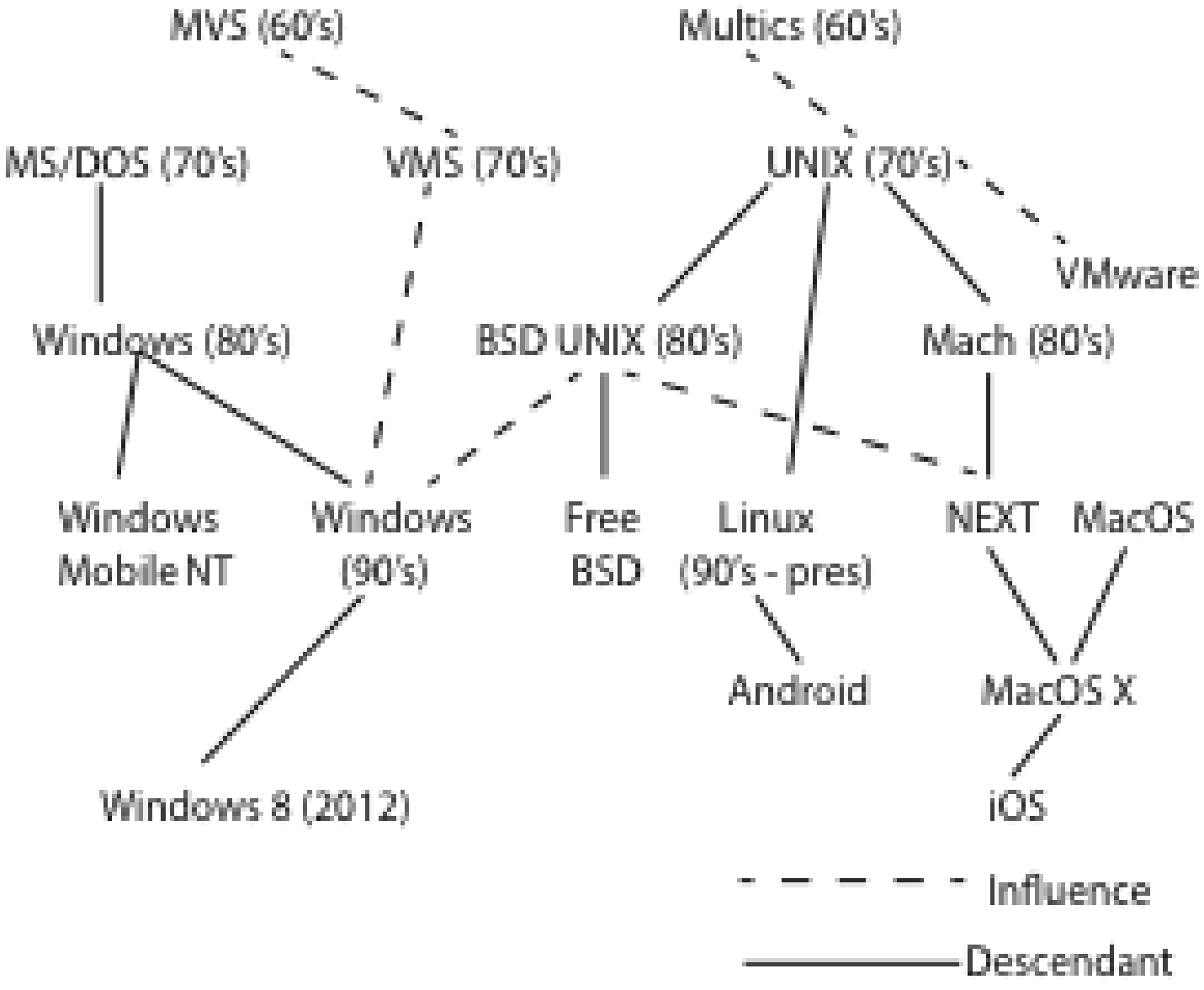
- Portability
  - For programs:
    - Application programming interface (API)
    - Abstract machine interface
  - For the operating system
    - Hardware abstraction layer



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


MVS (60's)
Multics (60's)
MS/DOS (70's)
VMS (70's)
UNIX (70's)
VMware
Windows (80's)
BSD UNIX (80's)
Mach (80's)
Windows Mobile NT
Windows (90's)
Free BSD
Linux (90's - pres)
NEXT
MacOS
Android
MacOS X
Windows 8 (2012)
iOS
Influence
Descendant

Computer Performance Over Time

	1981	1996	2011	factor
MIPS	1	300	10000	10K
MIPS/$	$100K	$30	$0.50	200K
DRAM	128KB	128MB	10GB	100K
Disk	10MB	4GB	1TB	100K
Home Internet	9.6 Kbps	256 Kbps	5 Mbps	500
LAN network	3 Mbps (shared)	10 Mbps	1 Gbps	300
Users per machine	100	1	<< 1	100+

Early Operating Systems: Computers Very Expensive

- One application at a time
  - Had complete control of hardware
  - "OS" was a 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
- Web servers
- Laptops
- Tablets
- Virtual machines
- …

Tomorrow's Operating Systems

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

Bonus Thought 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?
  - What if you need to allocate memory?
  - Disk?

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.”*