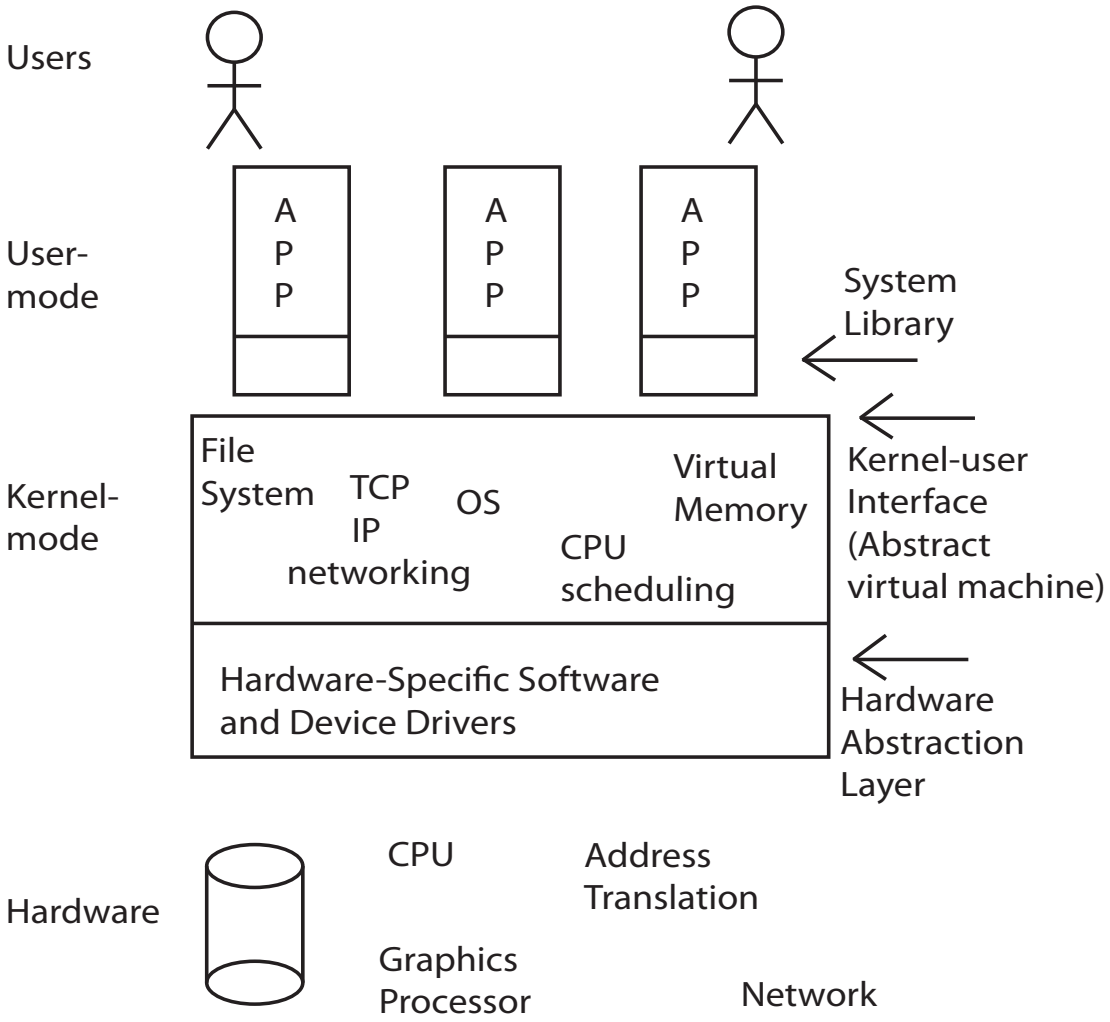
INTRODUCTION

Slides adapted from Tom Anderson's "Operating Systems: Principles and Practice"

What is an operating system?

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



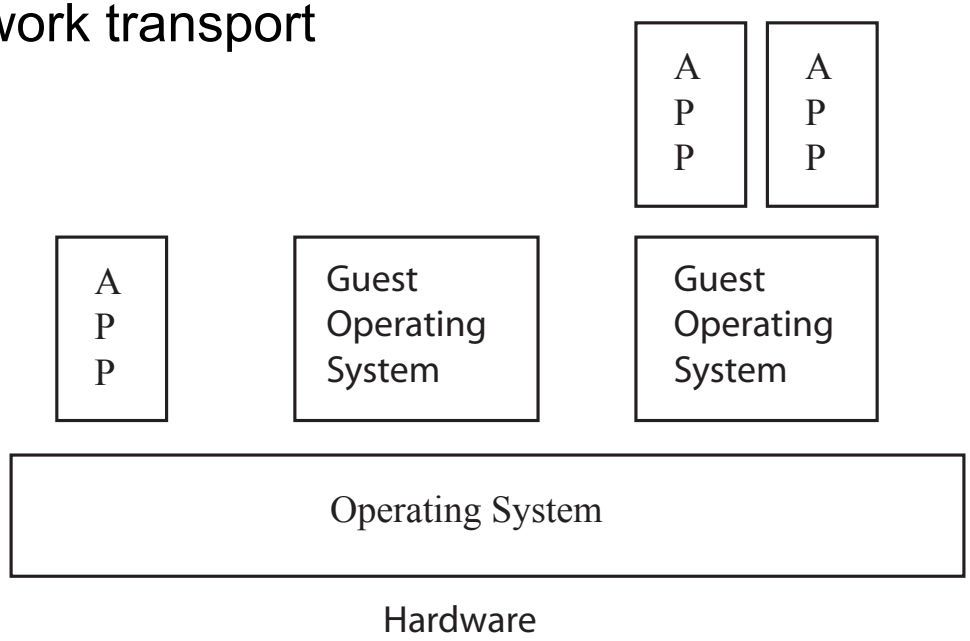
Slides adapted from Tom Anderson's "Operating Systems: Principles and Practice"

Operating System Roles

- Referee:
  - Resource allocation among users, applications
    - Choose how many resources must be awarded to each task
      - CPU, memory, …
  - Isolation of different users, applications from each other
    - Restrict the behavior of applications to less than the full power of the underlying hardware.
  - Communication between users, applications

Operating System Roles

- 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
  - Concept of virtualization
    - Virtualization of CPU, memory, devices
    - Virtualization of the entire computer → **Virtual Machines**
  - Mask other limitations of the hardware

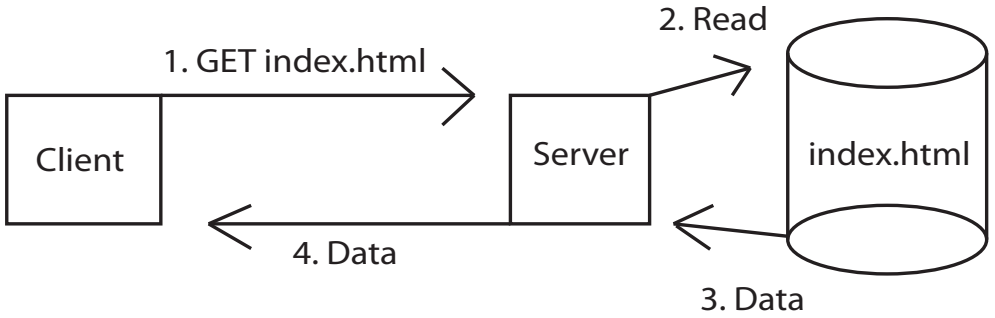


Operating System Roles

- Glue
  - Most of the OS code is glue code
  - Common services between applications
    - Libraries, user interface widgets, …

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?

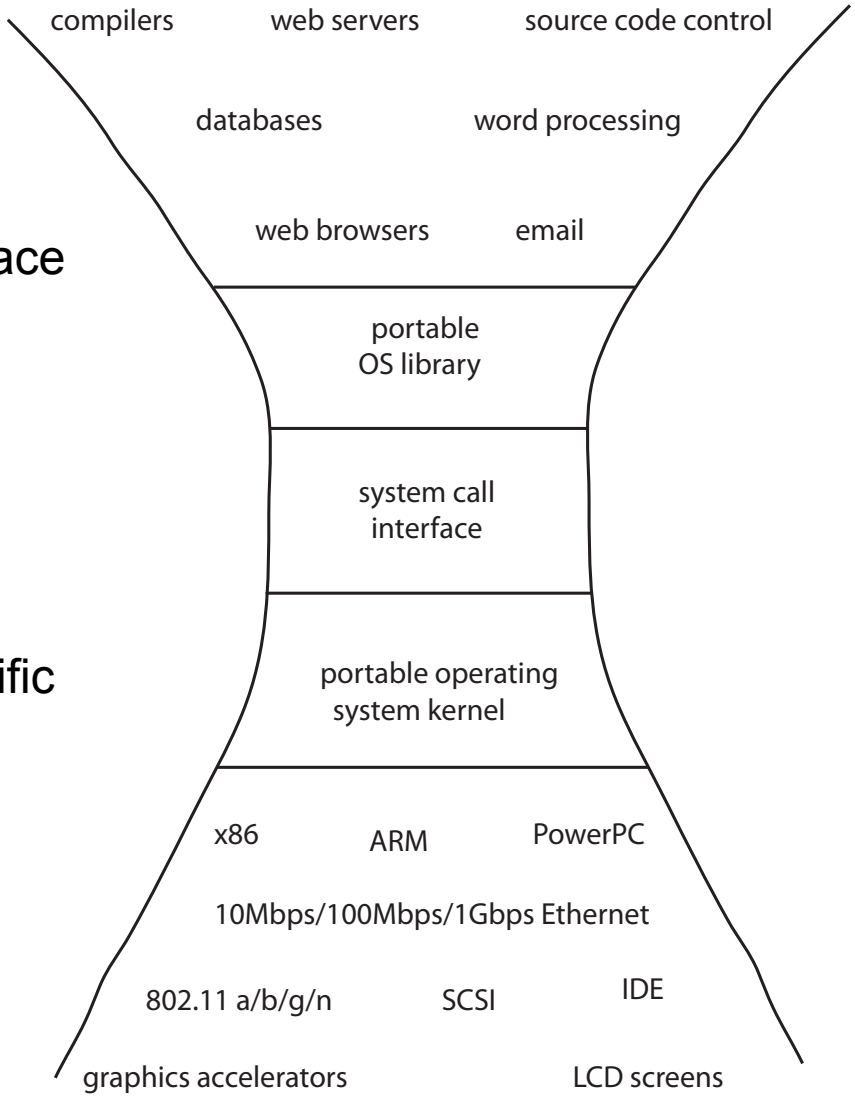


OS Challenges

- 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 (MTTR)
- 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
    - Pintos provides hardware-specific OS kernel routines

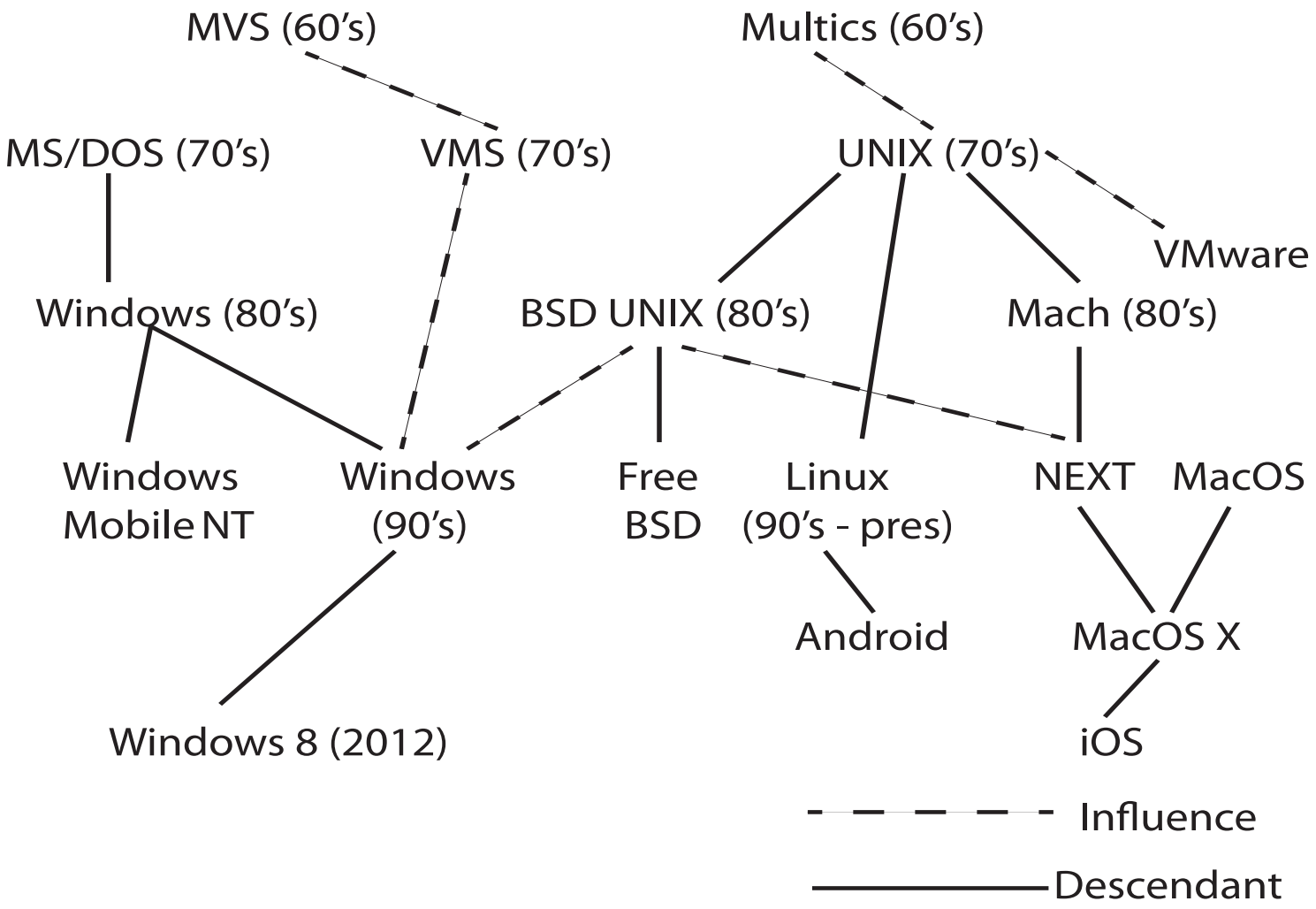


Slides adapted from Tom Anderson's "Operating Systems: Principles and Practice"

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



Slides adapted from Tom Anderson's "Operating Systems: Principles and Practice"

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 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

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?