Introduction to Operating Systems

CPSC/ECE 3220 Summer 2018

Lecture Notes
OSPP Chapter 1

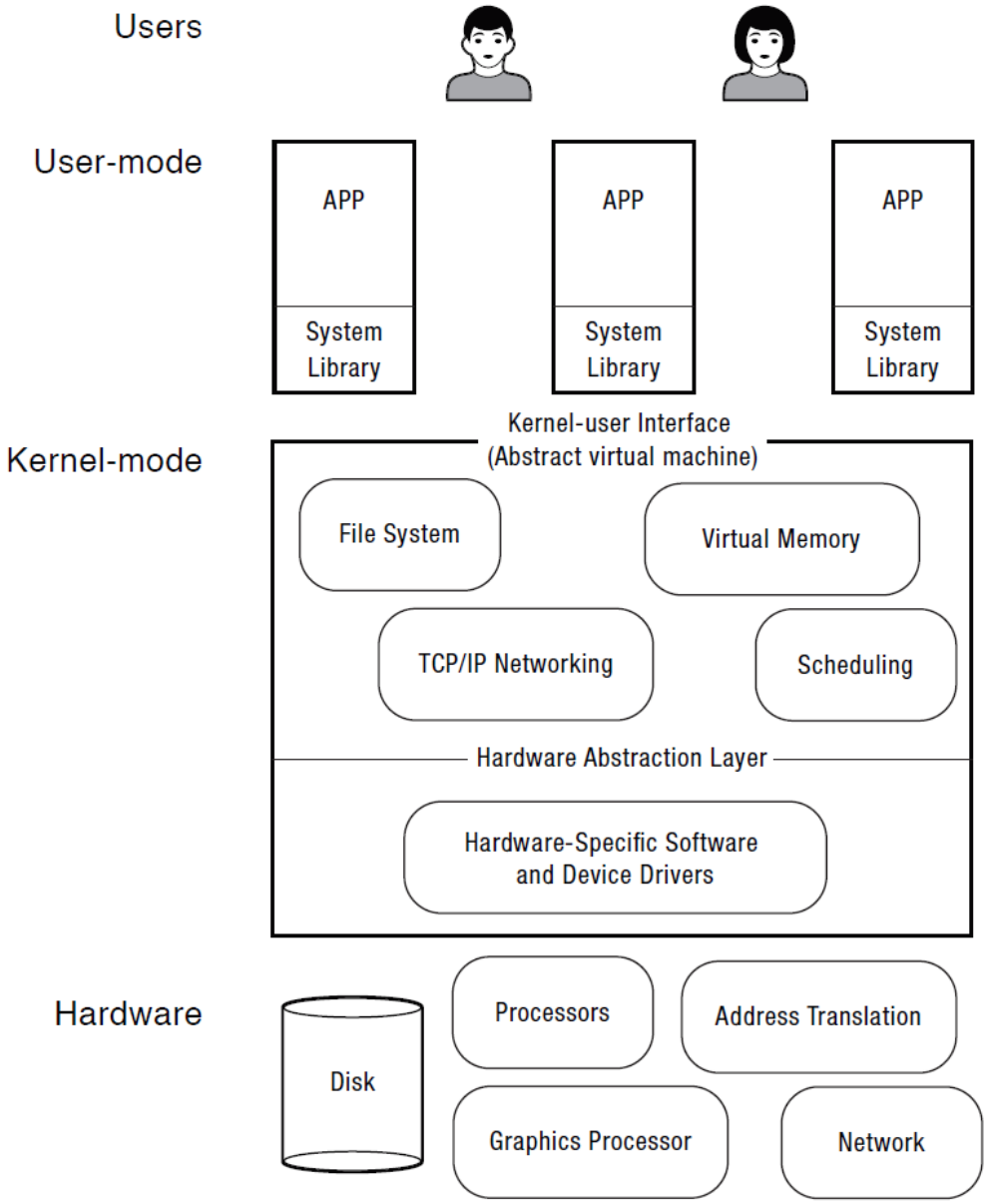
(adapted by Mark Smotherman from Tom Anderson's slides on OSPP web site)

Chapter 1

- Operating system definition
  - Software to manage a computer's resources for its users and applications
- OS roles
  - Referee, illusionist, and glue
- Core ideas
  - Protection, concurrency, resource allocation, virtualization, reliable storage

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 and applications
  - Isolation of users and applications from each other
  - Communication between users and applications
- Illusionist
  - Each application appears to have a machine to itself
  - Physical limitations and details are masked
  - Higher-level objects are provided, such as files
- Glue
  - Execution environment with common set of services
  - Files written by one app can be read by another

Example: File Systems

- Referee
  - Prevent users from accessing each other's files without permission
- Illusionist
  - Files and directories
  - Files can grow (nearly) arbitrarily large
  - Character or block I/O whether disk or network
  - Disk details such as sector size are hidden
- Glue
  - open(), fprintf(), fscanf()

Protection – Chapter 2

- The isolation of potentially misbehaving applications and users so that they do not corrupt other applications or the OS itself
- Prevent corruption of memory and files
- Prevent denial of service (DoS) to other users by unstoppable infinite loop on CPU
- Prevent crash of one application

Concurrency – Chapters 4-6

- Multiple activities that can happen at the same time
- Real concurrency: multiple CPUs
- Apparent concurrency: time sharing on a single CPU

Resource Allocation – Chapters 6-7

- Resource: a physical or virtual entity that can be assigned to a user or application
- E.g., OS decides how much CPU time and when, how much memory and when
- OS may limit allocations for purposes of efficiency and fairness
- OS controls the sharing of resources

Virtualization – Chapters 8-10

- Provide an application with the illusion of resources that are not physically present
- May be within a physical machine, such as virtual memory, or may be a full virtual machine

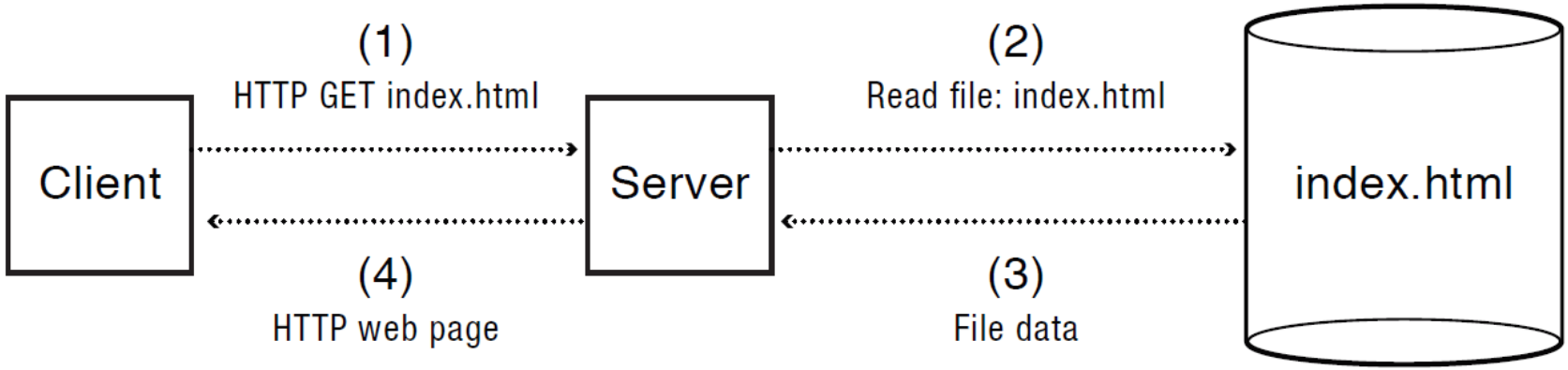
Reliable Storage – Chapters 11-14

- Safely store user data even if the system crashes due to software errors or hardware failures
- Atomically update multiple blocks of storage in a single transaction

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?
- Design choices represent trade-offs

Example: Web Service



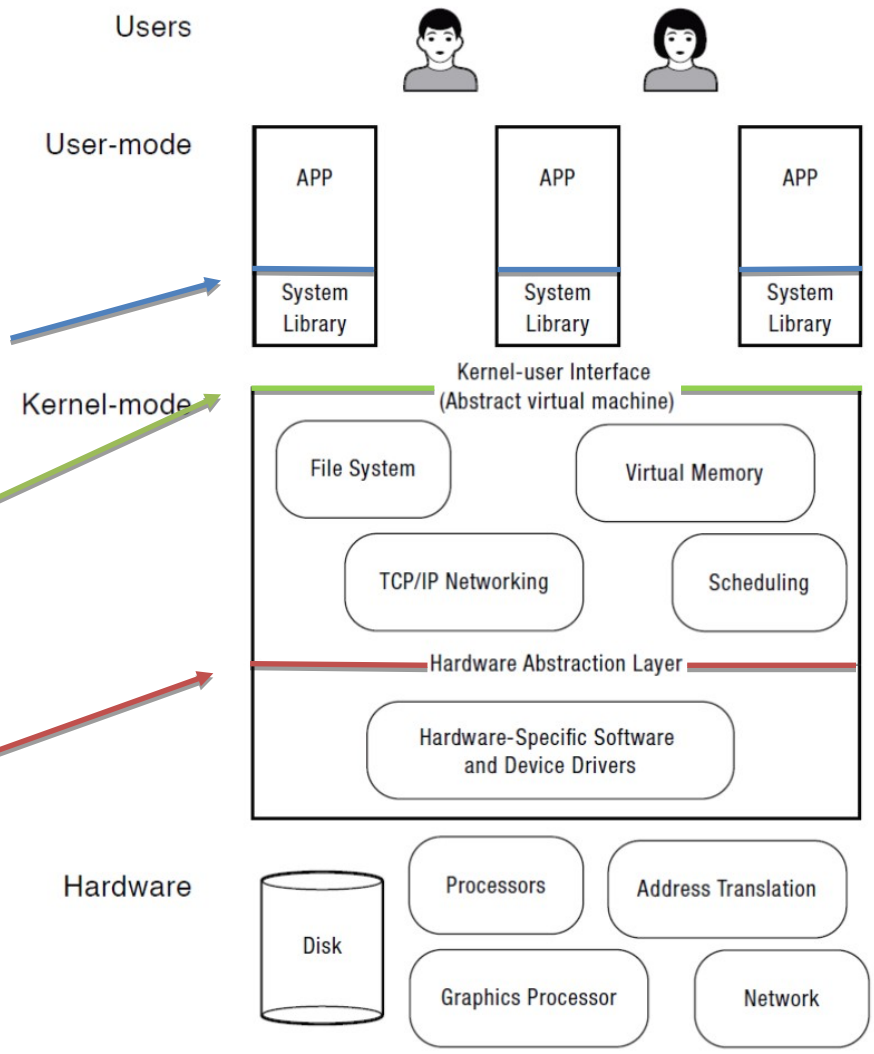
- How does the server manage many simultaneous client requests?
- How do we keep the client safe from spyware embedded in scripts on a web site?
- How do make updates to the web site so that clients always see a consistent view?

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 (MTTR)
    - Availability = MTTF / (MTTF + MTTR)
- Security
  - Can the system be compromised by an attacker?
- Privacy
  - Data is accessible only to authorized users

OS Challenges

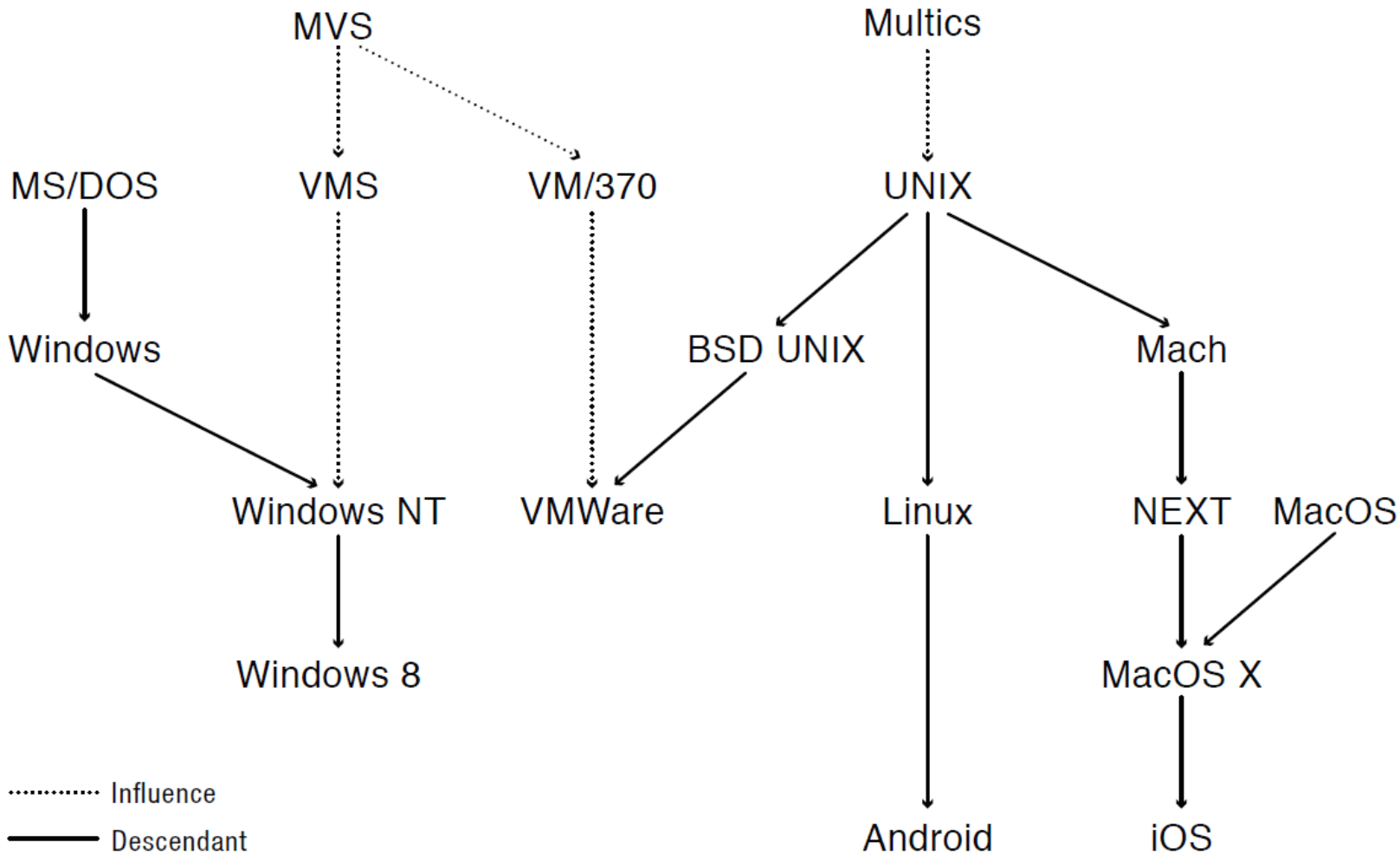
- Portability
  - For programs:
    - Application programming interface (API)
    - Abstract virtual machine (AVM)
  - For the operating system
    - Hardware abstractior layer



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?

Abbreviated OS History



Computer Performance Over Time

	1981	1997	2014	Factor (2014/1981)
Uniprocessor speed (MIPS)	1	200	2500	2.5K
CPUs per computer	1	1	10+	10+
Processor MIPS/$	$100K	$25	$0.20	500K
DRAM Capacity (MiB)/$	0.002	2	1K	500K
Disk Capacity (GiB)/$	0.003	7	25K	10M
Home Internet	300 bps	256 Kbps	20 Mbps	100K
Machine room network	10 Mbps (shared)	100 Mbps (switched)	10 Gbps (switched)	1000
Ratio of users to computers	100:1	1:1	1:several	100+

Early Operating Systems: Computers Were Very Expensive

- One application at a time
  - Had complete control of hardware
  - Runtime library was a primitive OS
  - 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

Time-Sharing Operating Systems: Computers and People Both 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 Are Cheap

- Smartphones
  - newer iPhones and iPads run two operating systems!
    - iOS, and
    - an Apple-customized version of the L4 microkernel on the Secure Enclave coprocessor
- 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