OPERATING SYSTEMS-1
CS 241
SPRING 2020

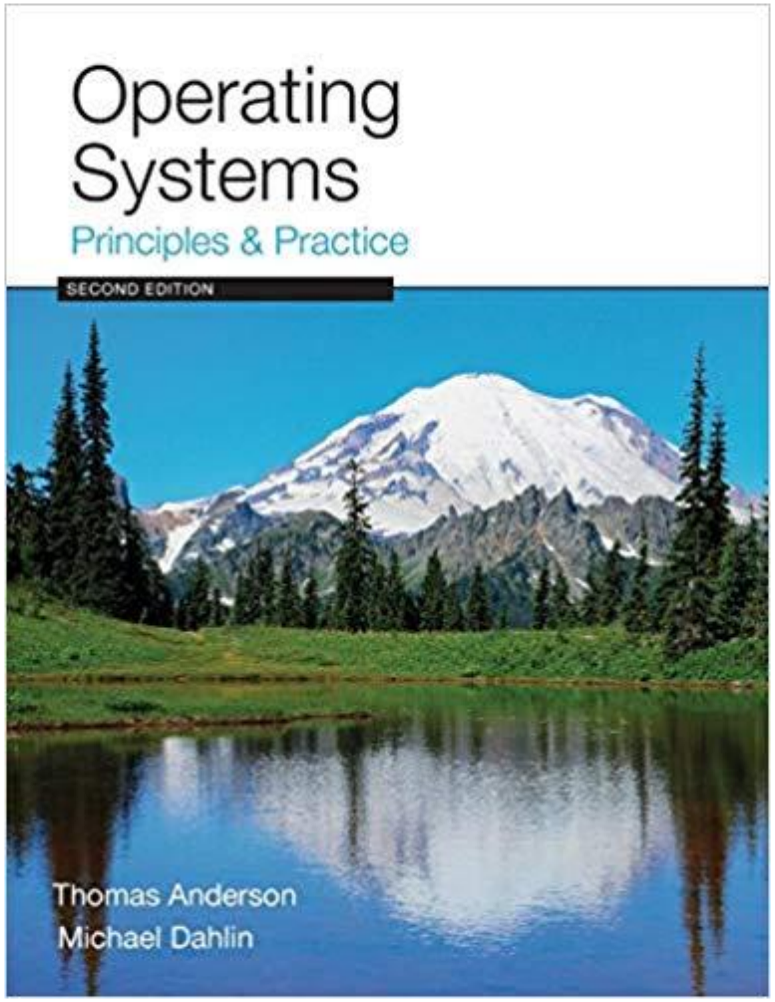
Ass. Prof. Ghada Ahmed
ghada_khoriba@fci.helwan.edu.eg

Edited slides,
main reference

Operating systems:
principles and practice, Tom Anderson, 2nd ed

Keep Your Soft Skills

Time Management
Networking
Teamwork
Respect Everybody's Ideas
Don't be LATE!
Creative Thinking
Conflict Resolution



OPERATING SYSTEMS: PRINCIPLES AND PRACTICE

TOM ANDERSON

Goals of this class

- Understand the structure and workings of an operating system (OS) –resource management

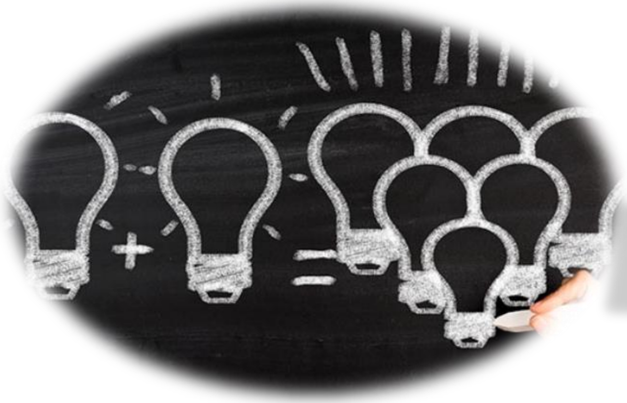
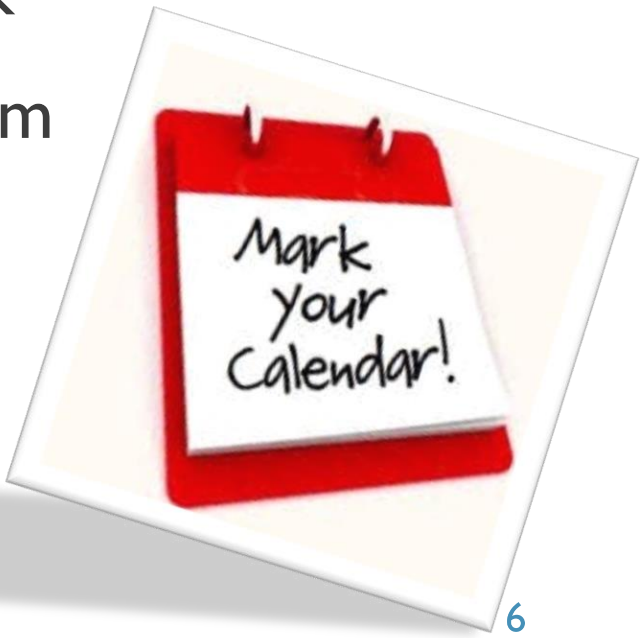
- Learn how to use and administrate the Linux OS
- Learn how to monitor your application, its environment and the hardware
- Understand the limits of hardware and how to design fast and reliable applications running on top of the OS

- After the class you should have a much **better understanding of the system software** that you a using indirectly through different applications.

WHY TO LEARN OPERATING SYSTEMS ??

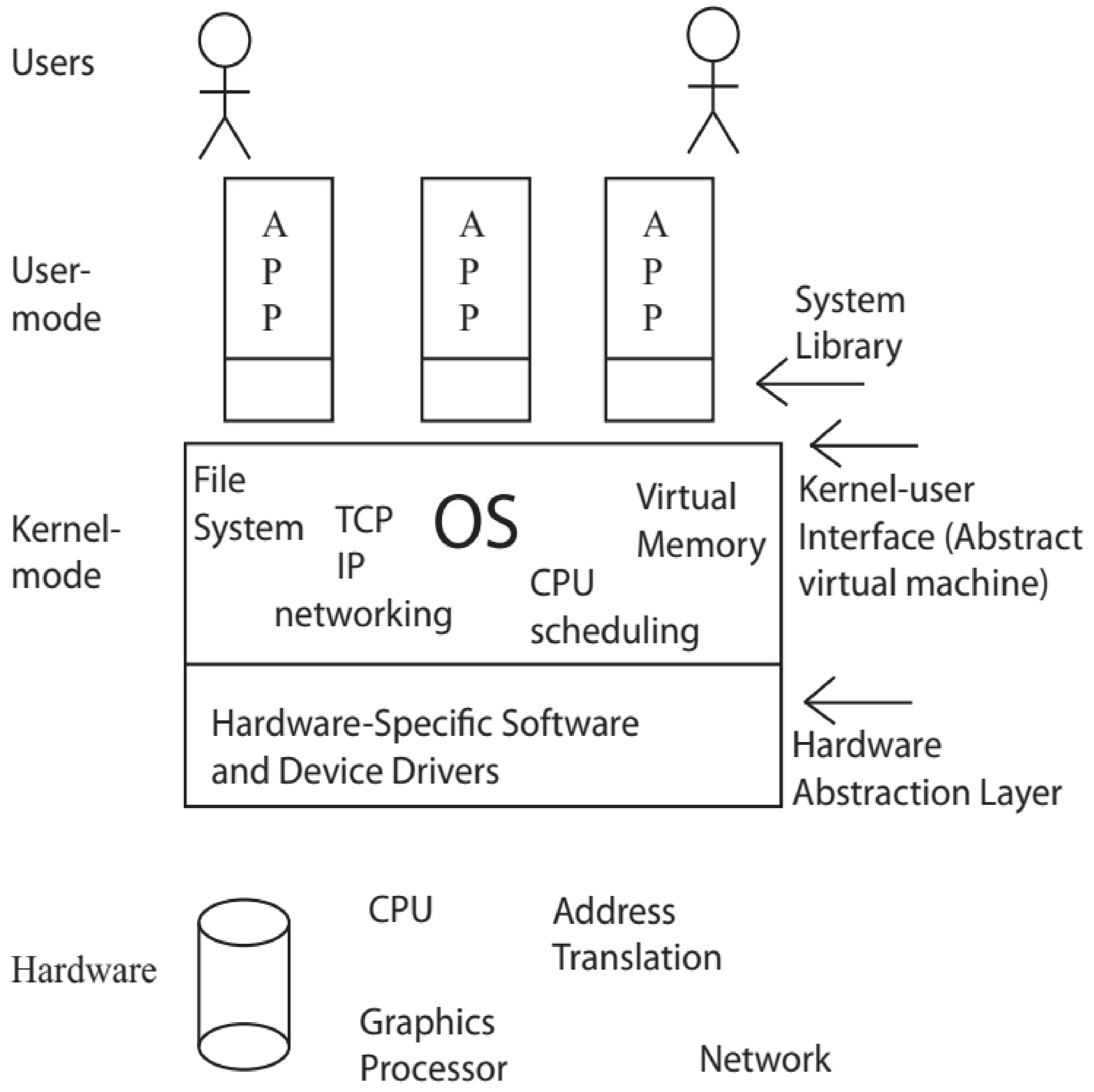
Grading

- 5 points online quiz …… 4th week
- 10 points for section tasks ….. 6th week & 9th week
- Midterm: 15 points ……. 7/8th week
- Project: 20 points ….. Practical exam
- Final exam: 50 points

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 , 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 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, …

QUESTION

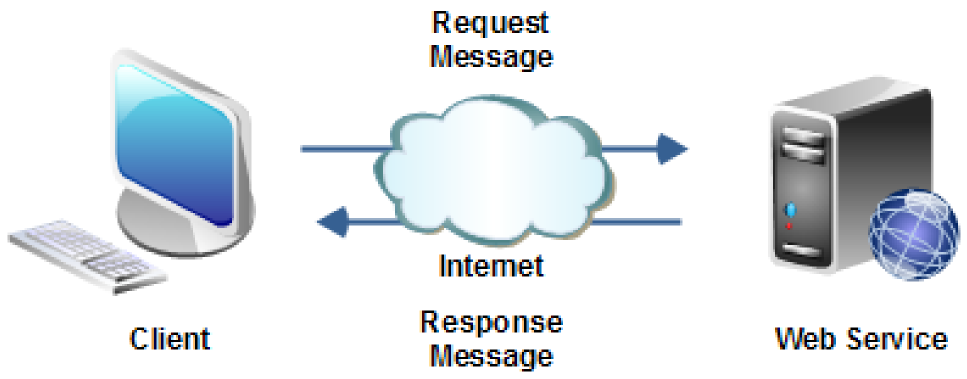
What (hardware, software) do you need to be able to run an **untrustworthy** application?

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

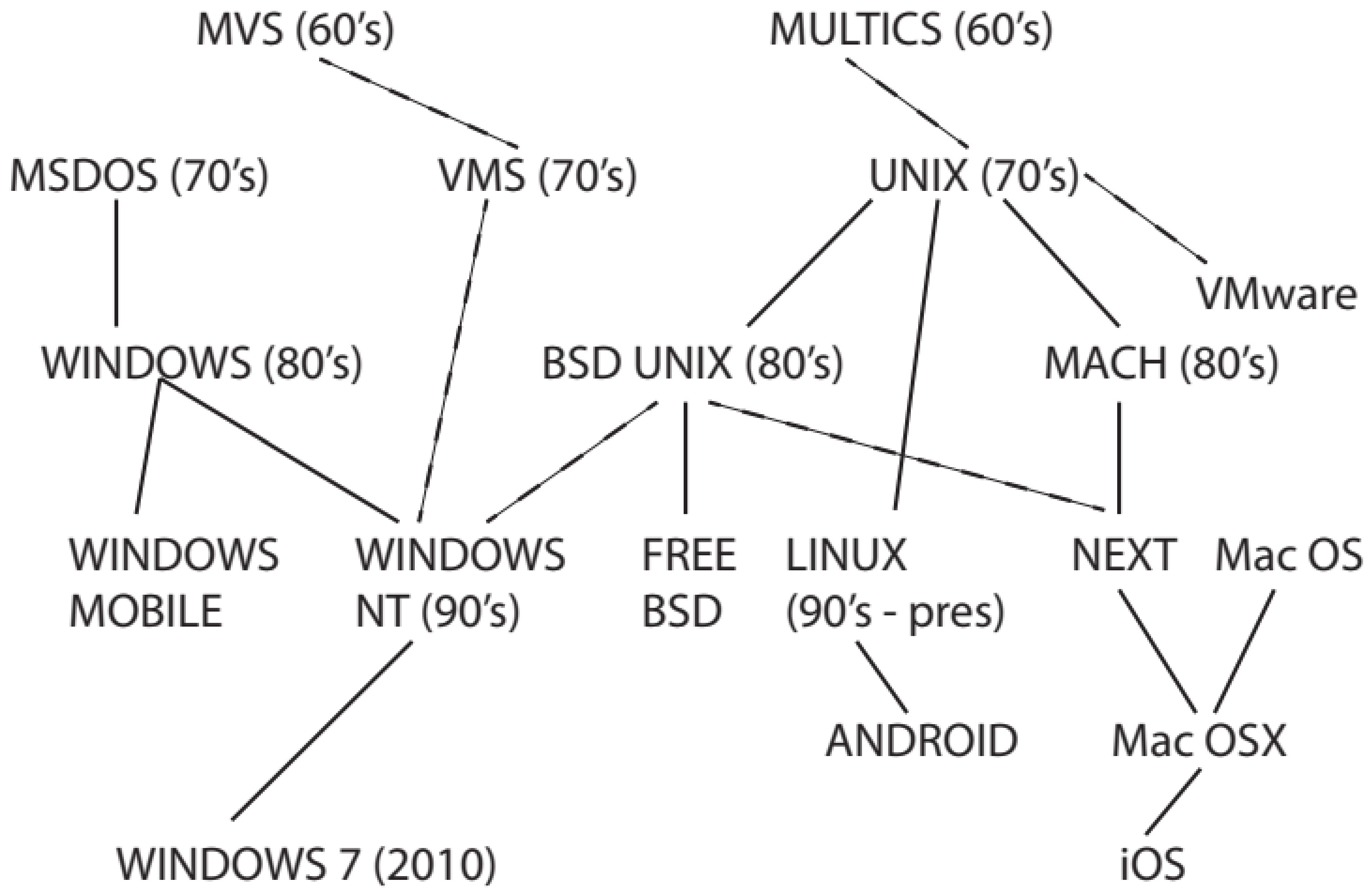
OS CHALLENGES

- 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


MVS (60's)
MULTICS (60's)
MSDOS (70's)
VMS (70's)
UNIX (70's)
VMware
WINDOWS (80's)
BSD UNIX (80's)
MACH (80's)
WINDOWS MOBILE
WINDOWS NT (90's)
FREE BSD
LINUX (90's - pres)
NEXT
Mac OS
ANDROID
Mac OSX
WINDOWS 7 (2010)
iOS

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

QUESTIONS?