Operating Systems

Lecture 1

Jinyang Li

Class goals

- Understand how an OS works by studying its:
  - Design principles
  - Implementation realities
- Gain some hands-on experience with implementing some OS components

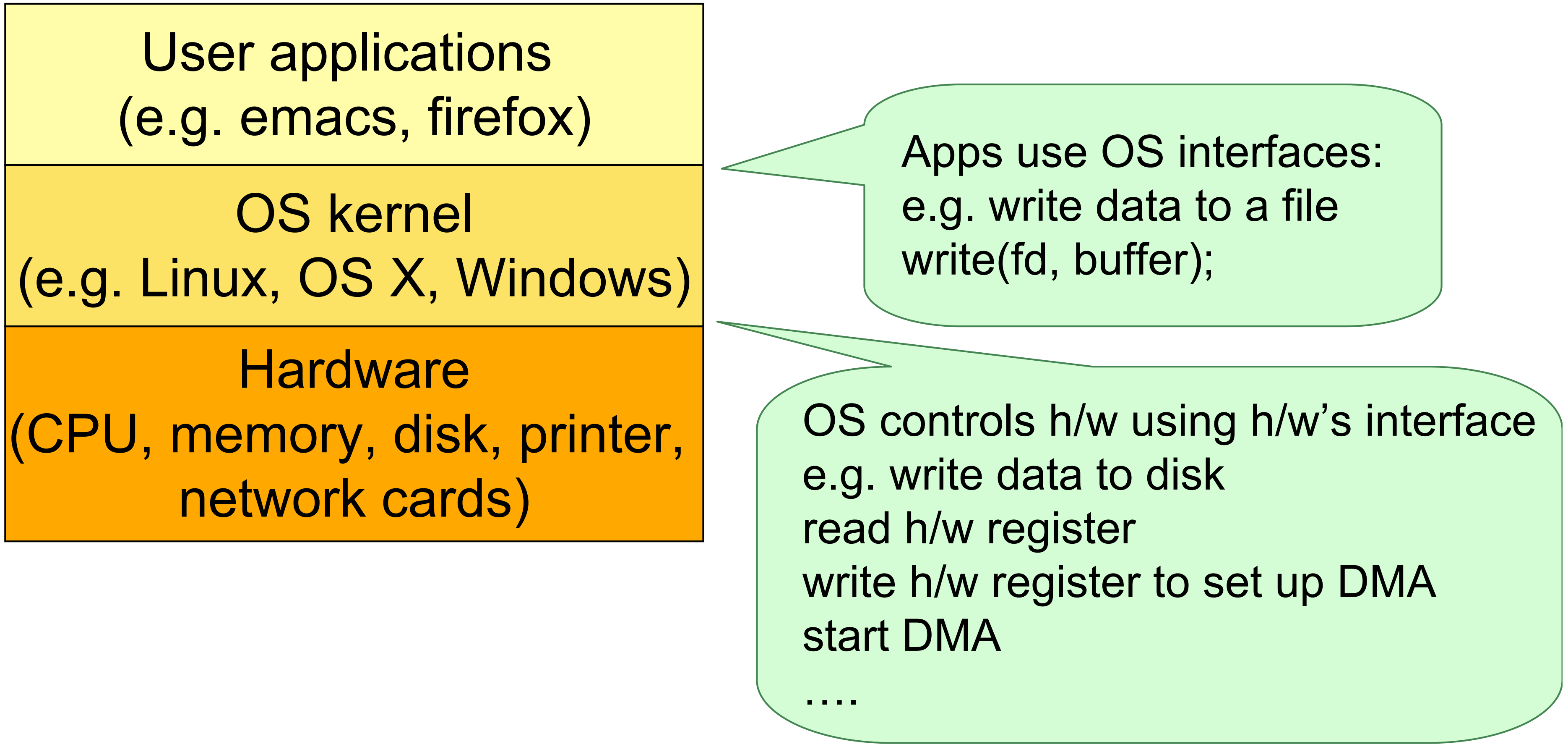
OS makes a computer “usable”

- What is an OS?
- Why need an OS? (Why not directly programm h/w?)
  - Lots of tedious details
  - Devices differ in their programming details
  - Redundant work across apps
  - Many apps want to use the same device (simultaneously)

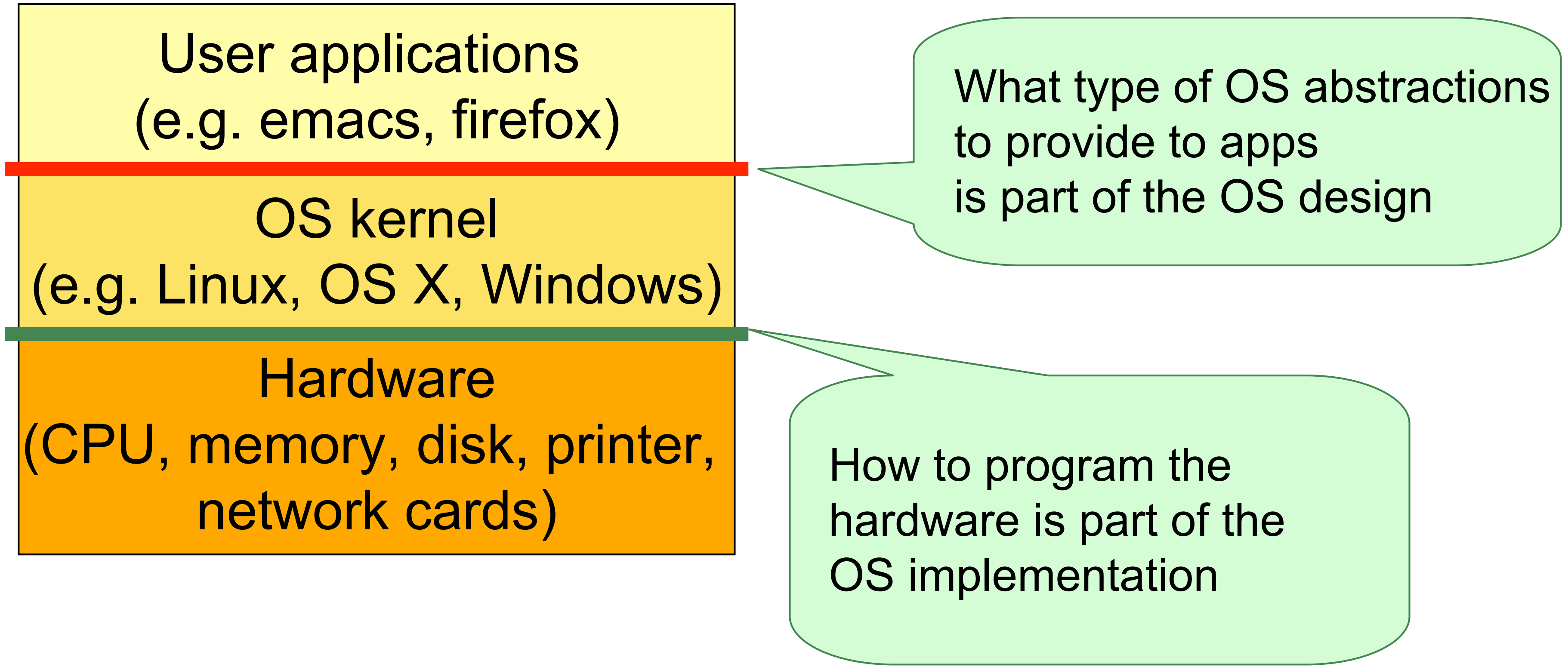
What’s an OS?

- The small view:
  - A library that manages hardware
- The big view:
  - OS provides an abstract machine on top of the physical machine
  - The abstract machine has better properties than the physical one

OS is an abstract machine


User applications
(e.g. emacs, firefox)
OS kernel
(e.g. Linux, OS X, Windows)
Hardware
(CPU, memory, disk, printer, network cards)
Apps use OS interfaces:
e.g. write data to a file
write(fd, buffer);
OS controls h/w using h/w's interface
e.g. write data to disk
read h/w register
write h/w register to set up DMA
start DMA
….

This class is about both *design & implementation*


User applications
(e.g. emacs, firefox)
OS kernel
(e.g. Linux, OS X, Windows)
Hardware
(CPU, memory, disk, printer, network cards)
What type of OS abstractions to provide to apps is part of the OS design
How to program the hardware is part of the OS implementation

Typical OS services

- Processes
- Address space
- File contents
- File namespace (directories, pathnames)
- Inter-process communication
- …

OS design goals

- Usability: abstract the hardware for programmer convenience
- Utilization: Multiplex the hardware among multiple applications
- Robustness: Isolate applications to contain bugs; prevent bad apps from crashing OS
- High Performance, low overhead

Why is OS design hard?

- Performance vs. ease of programmability
- Many features
- Complex component interactions
- Constantly evolving to handle new h/w, new app demands etc.
- Open problems: security, parallelism

Why should you take this class?

- OS is the foundation of systems programming
  - It’s challenging and important
- Learn what goes under the hood so you’ll be better at:
  - Using OS services effectively to build your own programs
  - Understanding the performance of your own programs
  - Diagnosing bugs and security problems

Class structure

- Staff:
  - Jinyang Li, 715 Broadway Rm 708
  jinyang@cs.nyu.edu
- Lectures
  - Teach basic OS concepts
  - Read required book chapters, papers, handouts before attending lecture
  - Check class schedule on the web often!

Class structure

- Mini-labs
  - 3 programming labs
  - C-based
  - Each lab is a fully functional small kernel illustrating one OS functionality
  - You need to have basic C programming experience to do the labs!
- Two quizzes
  - In-class quizzes, one mid-term, one final quiz
  - Strong students can skip final quiz and do a project instead (upon approval)

Class materials

- WebSite
  - http://www.news.cs.nyu.edu/~jinyang/sp09
  - Check schedule, announcements, reading preparation, labs etc.
- Textbook
  - ``Modern Operating Systems'' by Andrew Tanenbaum
  - Lectures loosely follow this book
- Class mailing list:
  - g22_2250_001_sp09@cs.nyu.edu
  - Email the list your question so everybody can contribute to the discussion

How are you evaluated?

- Class participation (10%)
- Three labs (15% each)
- Two quizzes (25% each)

Question?