UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 537
Intro to Operating Systems

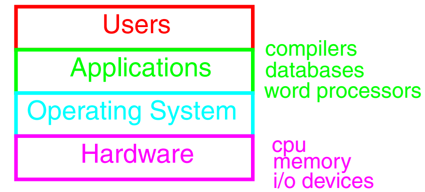
A. Arpaci-Dusseau
Spring 2000

Lecture 1: Introduction and Overview

- Questions answered in this lecture
  - What is an operating system?
  - How have operating systems evolved?
  - Why study operating systems?

What is an Operating System?

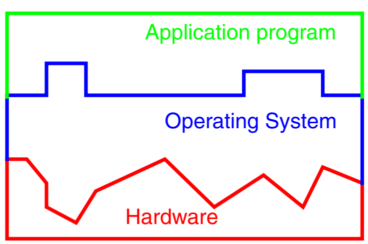
- Not easy to precisely define...



- OS:
  Everything in system that isn't an application or hardware
- OS:
  Software that converts hardware into useful form for applications
  - Standard library
  - Resource coordinator

First Function: Standard Library

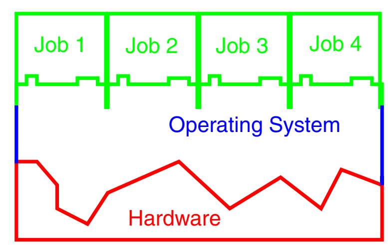
- Advantages of standard library
  - Allow applications to reuse common facilities
  - Make different devices look the same
  - Provide higher-level abstractions



- Challenges
  - What are the right abstractions?

Second Function: Resource Coordinator

- Resource: "Anything valuable" (e.g., CPU, memory, disk)
- Advantages of resource coordinator
  - Virtualize resource so multiple users/applications can share
  - Protect applications from one another
  - Provide efficient and fair access to resources



- Challenges
  - What mechanisms? What policies?

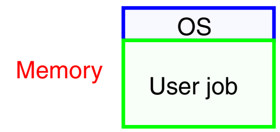
What Functionality in OS?

- No single right answer
  - Desired functionality depends on outside factors

- OS must adapt
  - Change abstractions provided to users
  - Change algorithms to implement those abstractions
  - Change low-level implementation to deal with hardware
- Current operating systems driven by its evolution
- Two distinct phases in history
  - Phase 1: Computers are expensive
  - Phase 2: Computers cheap

History of the OS

- First commercial systems
  - Enormous, expensive, and slow
  - Input/Output: Punch cards and line printers
- Goal: Get the system working
  - Single operator/programmer/user runs and debugs interactively
- OS Functionality
  - Standard library --> No coordination of resources
  - Monitor that is always resident; transfer control to programs



- Problem: Inefficient use of hardware
  - Performance metrics: **Throughput** and **utilization**

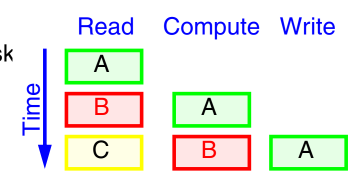
Batch Processing

- Batch: Group of jobs submitted to machine together
  - Operator collects jobs; orders efficiently; runs one at a time
- Role of OS: Same as before
- Advantages
  - Amortize setup costs over many jobs
  - Operator more skilled at loading tapes
  - Keep machine busy while programmer thinks
- Disadvantage
  - User must wait for results until batch collected and submitted

    (If bug, receive memory and register dump; submit job again!)
- Improve system throughput and utilization, but lose interactivity

Spooling

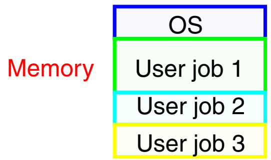
- Problem: Mechanical I/O devices much slower than CPU
  - Read 17 cards/sec vs. execute 1000s instructions/sec
- Spooling:
  Overlap I/O (e.g., card reading and line printing) with execution
  - Read card punches to disk
  - Compute (reading and writing to disk
  - Write output from disk to printer



- New OS functionality
  - Buffering, DMA, interrupt handling
- Advantage: Improves throughput and utilization
- Disadvantage: Single job must wait during I/O for data

Multiprogrammed Batch Systems

- Spooling provides pool of ready jobs
  - Keep multiple jobs resident in memory
  - OS chooses which job to run
  - When job waits for I/O, switch to another resident job



- New OS functionality
  - Job scheduling policies
  - Memory management and protection (virtual memory)
- Advantage: Improves throughput and utilization
- Disadvantage: Still not interactive

History of the OS: Phase 2

- Introduction of inexpensive, fast devices
  - Keyboards and monitors --> text editors and interactive debuggers
  - New set of performance trade-offs
- Goal: Improve user's response time

- Time-sharing:
  Switch between jobs to give appearance of dedicated machine

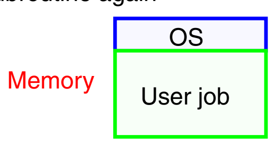
- Advantage
  - Users easily submit jobs and get immediate feedback

- New OS functionality
  - More complex job scheduling, memory management
  - Concurrency control and synchronization

Personal Computers

- Entire computer becomes inexpensive
  - Dedicated machine per user

- Remove functionality from OS
  - Remove time-sharing of multiple jobs
  - No protection
  - No virtual memory
  - OS becomes subroutine again



- Conclusion: OS Functionality changes with hardware and users

State of Current Systems

- Current trends
  - Multiprocessors
  - Networked systems
- Large
  - 100k's to millions of lines of code
  - 100-1000 person-years of work
- Complex
  - Performance is important
  - Conflicting needs of different users
- Poorly understood
  - System outlives any of its builders
  - Cannot remove all bugs
  - Behavior is hard to predict, tuning is done by guessing

Why Learn about Operating Systems?

- Tangible reasons
  - Build or modify a real operating system
  - Administer and use system well
  - Tune application performance

- Intangibles
  - Intrinsic curiosity
    - Understand how much of your computer system works
  - Gain/apply knowledge in other areas of Computer Science
    - Computer architecture and devices
    - Synchronization in programming languages
    - Data structures and algorithms
    - Performance analysis
  - Challenge of designing large, complex systems