EECS 482: Introduction to operating systems

Peter M. Chen
EECS Department
University of Michigan

A complete picture of programs execute?

• EECS 280, EECS 281
  – Ideas into high-level programming language
• EECS 370
  – High-level program into machine instructions
  – How processor executes machine instructions
• Missing pieces?
What’s an operating system?

• A software layer between the hardware and application programs

    application programs
    ┌───────────┐ ← virtual machine interface
    │           │
    └── operating system ┌───────────┘ ← physical machine interface
    └── hardware

• Creates abstractions to make hardware easier to use

• Manages shared hardware resources

• For any area of OS (e.g., threads, address spaces, file systems, networking, security), ask
  – What interface does hardware present to OS (physical reality)?
  – What interface does OS present to applications?

Relationship between applications and the OS

• Perspective 1: application is main program; it gets services by calling kernel (OS)

• Perspective 2: OS is main program; calls applications as subroutines
Why study operating systems?

• You may write part of one

• The purposes and techniques of an OS appear in many domains
  – Abstraction, management
  – Concurrency, caching, indirection, naming, atomicity, authentication, protection
  – Cloud computing, web servers, concurrent programs, virtual machine monitors, ...
  – OS principles are pervasive to all fields of computing

• Fun to “open the hood” and understand how things work

History of operating systems

• Hardware started out very expensive
• Operating systems started out very simple, then became more advanced to use expensive hardware more efficiently

• Single operator at console
  – Goal: basic functionality
  – Interactive
  – Very simple
    • One thing happening at a time
    • OS is library of standard services
  – Poor utilization of hardware resources
History of operating systems

• Batch processing
  – Goal: improve CPU and I/O utilization by removing user interaction
  – Submit job and wait for answer; no human interaction during execution
  – One job at a time
  – OS is batch monitor + library of standard services
  – Protection starts to become an issue: batch monitor must be able to run next program on queue
    • Why wasn’t this an issue for single operator at console?

• Multi-programmed batch
  – Goal: improve CPU and I/O utilization by overlapping CPU and I/O
  – Allows multiple I/Os to take place simultaneously
  – Allows CPU and I/O to take place simultaneously
  – OS getting more complex
    • OS switches between multiple processes
    • OS manages multiple I/O devices
    • OS must protect processes from each other
  – Still not interactive
History of operating systems

• Time sharing
  – Goal: allow people to interact with programs as they run
  – Insight: people can be modeled as a (very slow) I/O devices
  – Switch between processes while waiting for user

  – OS is now quite complicated
    • Lots of simultaneous jobs
    • Multiple sources of new jobs

• Personal computers
  – Is the driving assumption (hardware is expensive) still true?
  – How does this affect OS design?

  – Agree/disagree: personal computers don’t need to time share between multiple jobs?

  – Agree/disagree: personal computers don’t need protection between multiple jobs?

  – Personal computing operating systems have gradually added back all features from time-sharing systems
History of operating systems

• What’s next?
  – Operating systems for smartphones and wearable computing
  – Operating systems for cloud computing