CSE 120
Principles of Operating Systems

Fall 2000

Lecture 1: Course Introduction

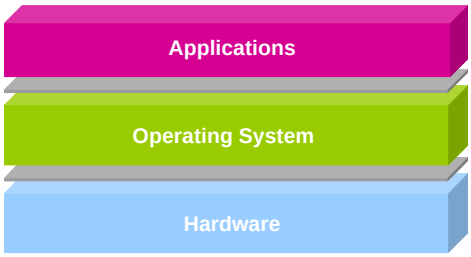
Geoffrey M. Voelker

Lecture 1 Overview

- Discuss handouts
- What is an operating system?
- Brief history of operating systems
- Class overview

What is an operating system?

- The operating system is the software layer between user applications and the hardware



- The OS is “all the code that you didn’t have to write” to implement your application

The OS and Hardware

- The OS abstracts/controls/mediates access to hardware resources
  - Computation (CPUs)
  - Volatile storage (memory) and persistent storage (disk, etc.)
  - Communication (network, modem, etc.)
  - Input/output devices (keyboard, display, printer, camera, etc.)
- The OS defines a set of logical resources (objects) and a set of well-defined operations on those objects (interfaces)
  - Physical resources (CPU and memory)
  - Logical resources (files, programs, names)

The OS and Hardware (2)

- Benefits to applications
  - Simpler (no tweaking device registers)
  - Device independent (all network cards look the same)
  - Portable (same program on Windows95/98/ME/NT/2000/…)
  - Transportable (same program across different OSes (Java))

The OS and Applications

- The OS defines a logical, well-defined environment…
  - Virtual machine (each program thinks it owns the computer)
- For users and programs to safely coexist, cooperate, share resources
  - Concurrent execution of multiple programs (timeslicing)
  - Communication among multiple programs (pipes, cut & paste)
  - Shared implementations of common facilities
    - » No need to implement the file system more than once
  - Mechanisms and policies to manage/share/protect resources
    - » File permissions (mechanism) and groups (policies)

OS Metaphors

- Service provider
  - The OS provides a standard set of facilities/services that enable programs to be simple and portable
- Executive/bureaucrat/big brother/juggler
  - The OS controls access to shared resources, and allocates resources for the greater good
- Caretaker
  - The OS monitors and recovers from exceptional conditions
- Cop/security guard
  - The OS mediates access to resources, granting or denying requests to use resources

Brief History of OS design

In the beginning…

- OSes were runtime libraries
  - The OS was just code you linked with your program and loaded into the computer
  - First computer interface was switches and lights, then punched tape and cards
- Batch systems were next
  - OS was permanently stored in primary memory
  - It loaded a single job (card reader, mag tape) into memory
  - Executed job, created output (line printer)
  - Loaded the next job, repeat…
  - Card readers, line printers were slow, and CPU was idle while they were being used

Spooling

- Disks provided a much faster I/O device than card readers, mag tape, and line printers
- Motivated development of spooling (Simultaneous Peripheral Operation On-Line)
  - Use disk to overlap I/O of one job with computation of others
  - Move program/data from card reader onto disk while another job computes
  - When done, next job ready to be loaded from disk
  - Can spool multiple programs onto disk, OS can choose which job to run next (job scheduling)
  - But, CPU still idle when job reads/writes to disk

Multiprogramming

- Multiprogramming increased system utilization
  - Keeps multiple runnable jobs loaded in memory
  - Overlaps I/O processing of a job with computation of another
  - Benefits from I/O devices that can operate asynchronously
  - Requires the use of interrupts and DMA
  - Optimizes system throughput (number of jobs finished in a given amount of time) at the cost of response time (time until a particular job finishes)

Timesharing

- Timesharing supports interactive use of computer by multiple users
  - Terminals give the illusion that each user has own machine
  - Optimizes response time (time to respond to an event like a keystroke) at the cost of throughput
  - Based on timeslicing – dividing CPU time among the users
  - Enabled new class of applications – interactive!
  - Users now interact with viewers, editors, debuggers
- The MIT Multics system (mid-late 60s) was an early, aggressive timesharing system
- Unix is also a timesharing system …what about Windows?

Distributed Operating Systems

- Distributed systems facilitate use of geographically distributed resources
  - Machine connected by wires
- Supports communication between parts of a job or different jobs on different machines
  - Interprocess communication
- Sharing of distributed resources, hardware, and software
  - Exploit remote resources
- Enables parallelism, but speedup is not the goal
  - Goal is communication

Parallel Operating Systems

- Support parallel applications trying to get speedup of computationally complex tasks across multiple CPUs
- Requires basic primitives for dividing single task into multiple parallel activities
- Supports efficient communication among activities
- Supports synchronization of activities to coordinate data sharing
- Early parallel systems used dedicated networks and custom CPUs, now common to use networks of high-performance PC/workstations

Embedded Operating Systems

- Decreased cost of processing makes computers ubiquitous
  - Your car has dozens of computers in it
  - Think of everything that has electric motor in it, and now imagine that it also has a computer
- Each embedded application needs its own OS
  - Cell phones
  - PDAs (PalmPilot, etc.)
- Very soon
  - Your house will have 100s of embedded computers in it
  - Your electrical lines and airwaves will serve as the network
  - All devices will interact as a distributed system

CSE 120

- This course addresses classic OS concepts
  - The services provided by the OS
  - OS implementation on modern hardware
  - Co-evolution of hardware and software
  - Techniques for implementing software systems that are
    - » Large and complex
    - » Long-lived and evolving
    - » Concurrent
    - » Performance-critical
  - Remove all mysteries

Fundamental OS Issues

- The fundamental issues/questions in this course are:
  - Structure: how is an operating system organized?
  - Sharing: how are resources shared among users?
  - Naming: how are resources named (by users and programs)?
  - Protection: how are users/programs protected from each other?
  - Security: how can information access/flow be restricted?
  - Communication: how to exchange data?
  - Reliability and fault tolerance: how to mask failures?
  - Extensibility: how to add new features?

Fundamental OS Issues (2)

- Concurrency: how to control parallel activities?
- Performance: how to make efficient use of resources, reduce OS overhead?
- Scale and growth: how to handle increased demand?
- Compatibility: can we ever do anything new?
- Distribution: how to coordinate remote operations?
- Accountability: how to charge for/restrict use of resources?

- And the "principles" in this course are the methods/approaches/solutions to these issues

Course Components

- Course material provided through class lectures, textbook readings, and handouts
- Course assignments are homework questions from the book and a series of class projects
  - Approximately five spaced throughout the quarter
- Discussion sections are a forum for asking questions
  - Will have mailing list and online discussion forums, too

Nachos

- Nachos is an instructional operating system
  - It is a user-level operating system and a machine simulator
    - » Not necessarily unlike the Java runtime environment
    - » Will become abundantly clear (or not so clear) very soon
  - Written in C++
  - Programming environment will be Unix (Linux)
  - The projects will require serious time commitments
- There will be three projects using Nachos
  - Concurrency and synchronization
  - Virtual memory
  - File system
- You will work in groups of four on the projects

For next class...

- Browse the course web (if it ever gets fixed…)
  - http://www-cse.ucsd.edu/classes/fa00/cse120/
- Read Chapters 1 and 2
- Send your email address to me (voelker@cs.ucsd.edu) for mailing list
- Start thinking about partners for project groups

- The lab machines are being transitioned to Linux (they have NT on them)
- I will bring account slips at the next lecture (not ready before class today)