1. Introduction

1.1 The Role of Operating Systems
   - Bridge the “Semantic Gap” between Hardware and Application
   - Three Views of Operating System
     1. Abstraction (addresses complexity)
     2. Virtualization (addresses sharing)
     3. Resource management (addresses performance)

1.2 Organization of Operating Systems
   - Structural Organization
   - The Hardware Interface
   - The Programming Interface
   - The User Interface
   - Runtime Organization

1.3 Operating System Evolution & Concepts
Single CPU System

![Diagram of a single CPU system with components including Main memory, CPU, Communication devices, Secondary storage, User, and Network.]

Figure 1-1
Multiprocessor Systems

Figure 1-2a

Figure 1-2b
Multicomputer System

Figure 1-2c
PC Hardware Organization

Figure 1-7
Bridging the Semantic Gap

• Hardware capabilities are very low level
  – Arithmetic and logical operators
  – Comparison of two bit-strings
  – Branching, reading, and writing bytes

• User needs to think in terms of problem to be solved
  – High-level data structures and corresponding operations
  – Simple, uniform interfaces to subsystems,
  – Treat programs and data files as single entities

• Use software to bridge this gap
  – Language processors (e.g., assemblers, compilers, interpreters).
  – Editors and text processors, linkers and loaders.
  – Application programs, utility and service programs.
  – **Operating Systems**
The role of OSs

• **Bridge Hardware/Application Gap**
  – Machine instruction *vs.* high level operation
    • compiler bridges gap
  – Linear memory *vs.* data structures
    • compiler bridges gap
  – Limited CPU & memory *vs.* more needed
    • OS bridges gap
  – Secondary memory devices *vs.* files
    • OS bridges gap
  – I/O devices *vs.* high level I/O commands
    • OS bridges gap
Three views of OSs

- **OS is an extended machine**
  - Principle of abstraction hides complexity
  - OS provides high level operations using lower level operations

- **OS is a virtual machine**
  - Principle of virtualization supports sharing
  - OS provides virtual CPU, memory, devices

- **OS is a resource manager**
  - Balance overall performance with individual needs (response time, deadlines)
Structural Organization of OSs

- Monolithic vs. Layered

Figure 1-8
Organization of OSs

• Hardware Interface
  – Applications and OS compiled into machine instructions
  – Interrupts and Traps allow OS to seize control
    • process management (time-sharing)
    • device management (I/O completion)
Interrupts vs. Traps

![Diagram of Interrupts vs. Traps]

(a) External hardware device

(b) Cause trap

Figure 1-9

CompSci 143A
Spring, 2013
Organization of OSs

• Hardware interface (continued)
  – Modes of CPU execution
    • Privileged/Nonprivileged
    • SVC (supervisor call) causes trap
    • Control transferred to OS in privileged mode
    • OS exits privileged mode when returning to user
Organization of OSs

- **Programming Interface**

- **Invoking system services**
  - Library call (nonprivileged)
  - Kernel call (privileged)

Figure 1-8
Invoking System Services

(a)

(b)

Figure 1-10
Organization of OSs

• User interface (cf. Fig. 1-8)
  – Text-based shell (e.g. Unix)
    • command interpreter
    • shell scripts
  – Graphics-based GUI (e.g. Mac, MS Windows)
    • Windows
    • Icons
    • Menus
    • Pointer
Organization of OSs

- Runtime organization
  - Service is a Subroutine
  - Service is an Autonomous Process ("client-server")

Figure 1-12
OS Evolution and Concepts

• Early systems
  – Bootstrapping

• Batch OSs
  – I/O processors
  – Interrupts
  – Relocatable code (Allows separately compile programs to be combined without reccompilation)

• Multiprogramming
Multiprogramming

• Basic problem:
  – Some programs are compute-bound, some I/O-bound
  – Even “balanced” programs are balance only over time
  – No one program can make full use of the system

• Solution: Multiprogramming
  – Have more than one active (running) program in memory at any one time

• Multiprogramming requires
  – Bridging the semantic gap
  – Sharing resources among different programs
  – Hiding from each program the fact of this sharing
OS Evolution and Concepts

• Multiprogramming Systems
  – Overlap CPU and I/O
  – Protection
  – Synchronization and Communication
  – Dynamic Memory Management
    (swapping and paging)

• Interactive OSs
  – Guaranteed response time
  – Time-sharing (quantum)
OS Evolution and Concepts

• PC and workstation OSs
  – GUI

• Real-time OSs
  – Deadlines (scheduling)

• Distributed OSs
  – Loosely coupled/tightly coupled
  – Consistent timeline (logical clocks, time stamps)
Abstraction

• E.W. Dijkstra, “The Humble Programmer” (1972): “The purpose of abstraction is *not* to be vague, but to create a new semantic level in which one can be absolutely precise.”

• “abstraction” is created by distinguishing
  – Essential characteristics from Unimportant details

• Build levels (layers) of abstractions:
  – What is unimportant detail at one level is an essential characteristic at a lower one.
History
• Originally developed by Steve Franklin
• Modified by Michael Dillencourt, Spring, 2009
• Modified by Michael Dillencourt, Summer 2012