Operating System Overview

Chapter 1.5 – 1.9
Learning Outcomes

• A high-level understanding of the structure of operating systems, applications, and the relationship between them.
• Some knowledge of the services provided by operating systems.
• Exposure to some details of major OS concepts.
Operating System

• A program that controls execution of applications
  – The resource manager
• An interface between applications and hardware
  – The extended machine
Structure of a Computer System

User Mode

Application

System Libraries

Application

System Libraries

Application

System Libraries

Kernel Mode

Operating System

Hardware
Structure of a Computer System

User Mode

Application

System Libraries

Kernel Mode

Operating System

Hardware

Interacts via load and store instructions to CPU and device registers, and interrupts
Structure of a Computer System

User Mode

Application

Interaction via function calls to library procedures

System Libraries

Kernel Mode

Operating System

Hardware
Structure of a Computer System

User Mode

Application

System Libraries

Interaction via

System Calls

Kernel Mode

Operating System

Hardware
A note on System Libraries

- System libraries are just that, libraries of support functions (procedures, subroutines)
  - Only a subset of library functions are actually systems calls
    - strcmp(), memcpy(), are pure library functions
    - open(), close(), read(), write() are system calls
  - System call functions are in the library for convenience
Operating System Objectives

• Convenience
  – Make the computer more convenient to use

• Abstraction
  – Hardware-independent programming model

• Efficiency
  – Allows the computer system to be used in an efficient manner

• Ability to evolve
  – Permit effective development, testing, and introduction of new system functions without interfering with existing services

• Protection
Services Provided by the Operating System

- Program development
  - Editors, compilers, debuggers
    - Not so much these days

- Program execution
  - Load a program and its data

- Access to I/O devices

- Controlled access to files
  - Access protection

- System access
  - User authentication
Services Provided by the Operating System

• Error detection and response
  – internal and external hardware errors
    • memory error
    • device failure
  – software errors
    • arithmetic overflow
    • access forbidden memory locations
  – operating system cannot grant request of application
Services Provided by the Operating System

• Accounting
  – collect statistics
  – monitor performance
  – used to anticipate future enhancements
  – used for billing users
Operating System Software

- Fundamentally, OS functions the same way as ordinary computer software
  - It is a program that is executed (just like apps)
  - It has more privileges
- Operating system relinquishes control of the processor to execute other programs
  - Reestablishes control after
    - System calls
    - Interrupts (especially timer interrupts)
Kernel

- Portion of the operating system that is running in *privileged mode*
- Usually resident in main memory
- Contains fundamental functionality
  - Whatever is required to implement other services
  - Whatever is required to provide security
- Contains most-frequently used functions
- Also called the nucleus or supervisor
Major OS Concepts

- Processes
- Concurrency and deadlocks
- Memory management
- Files
- Information Security and Protection
- Scheduling and resource management
Processes

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of resource ownership
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources
  - Nowadays the execution abstraction is separated out: *Thread*
  - Single process can contain many threads
Process

- Consist of three segments
  - Text
    - contains the code (instructions)
  - Data
    - Global variables
  - Stack
    - Activation records of procedure
    - Local variables
- Note:
  - data can dynamically grow up
  - The stack can dynamically grow down

Memory

- Stack
- Gap
- Data
- Text
Process

- Consists of three components
  - An executable program
    - text
  - Associated data needed by the program
    - Data and stack
  - Execution context of the program
    - All information the operating system needs to manage the process
      - Registers, program counter, stack pointer, etc…
    - A multithread program has a stack and execution context for each thread
Multiple processes creates concurrency issues

(a) A potential deadlock. (b) an actual deadlock.
Memory Management

- The view from thirty thousand feet
  - Process isolation
    - Prevent processes from accessing each other’s data
  - Automatic allocation and management
    - Don’t want users to deal with physical memory directly
  - Support for modular programming
  - Protection and access control
    - Still want controlled sharing
  - Long-term storage
  - OS services
    - Virtual memory
    - File system
Virtual Memory

• Allows programmers to address memory from a logical point of view
  – Gives apps the illusion of having RAM to themselves
  – Logical addresses are independent of other processes
  – Provides isolation of processes from each other

• Can overlap execution of one process while swapping in/out others.
Virtual Memory Addressing

Figure 2.10  Virtual Memory Addressing
File System

- Implements long-term store
- Information stored in named objects called files
Example File System
Information Protection and Security

- **Access control**
  - regulate user access to the system
  - Involves authentication

- **Information flow control**
  - regulate flow of data within the system and its delivery to users
Scheduling and Resource Management

- **Fairness**
  - give equal and fair access to all processes

- **Differential responsiveness**
  - discriminate between different classes of jobs

- **Efficiency**
  - maximize throughput, minimize response time, and accommodate as many uses as possible
Operating System Structure

- The layered approach
  a) Processor allocation and multiprogramming
  b) Memory Management
  c) Devices
  d) File system
  e) Users

  Each layer depends on the inner layers
Operating System Structure

- In practice, layering is only a guide
  - Operating Systems have many interdependencies
    - Scheduling on virtual memory
    - Virtual memory on I/O to disk
    - VM on files (page to file)
    - Files on VM (memory mapped files)
    - And many more…
The Monolithic Operating System Structure

- Also called the “spaghetti nest” approach
  - Everything is tangled up with everything else.
- Linux, Windows,
The Monolithic Operating System Structure

- However, some reasonable structure usually prevails
UNIX

• Provides a good hardware abstraction
  – Everything is a file (mostly)
• Runs on most hardware
• Comes with a number of user services and interfaces
  – shell
  – C compiler
Traditional UNIX Structure

UNIX Commands and Libraries

System Call Interface

Kernel

Hardware

User-written Applications

Figure 2.15 General UNIX Architecture
Traditional UNIX Kernel
Microkernel-based Systems

• Assigns only a few essential functions to the kernel
  – Address space
  – Interprocess Communication (IPC)
  – Basic scheduling
  – Minimal hardware abstraction

• Other services implemented by user-level servers

• Traditional “system calls” become IPC requests to servers

• Extreme view of a microkernel
  – A feature is only allowed in the kernel if required for security
Application

Monolithic Kernel

Bit  Byte  Word  Register  Instructions  HW