Chapter 1: Introduction
Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Kernel Data Structures
- Computing Environments
- Open-Source Operating Systems
Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware

- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner
Computer System Structure

Computer system can be divided into four components:

- **Hardware** – provides basic computing resources
  - CPU, memory, I/O devices
- **Operating system**
  - Controls and coordinates use of hardware among various applications and users
- **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- **Users**
  - People, machines, other computers
Four Components of a Computer System

user 1
user 2
user 3
...
user n

compiler
assembler
text editor
...
database system

system and application programs

operating system

computer hardware
What Operating Systems Do

- Depends on the point of view
- PC users want *convenience, ease of use* and *good performance*
  - Don’t care about *resource utilization*
- But shared computer such as *mainframe* or *minicomputer* must keep all users happy, and thus resource utilization must be efficient.
- Users of dedicate systems such as *workstations* have dedicated resources but frequently use shared resources from *servers*
  - Balance between individual usability and resource utilization
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles
Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use

- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer
Operating System Definition (Cont.)

- No universally accepted definition

- “Everything a vendor ships when you order an operating system” is a good approximation
  - But varies wildly

- “The one program running at all times on the computer” is the kernel.

- Everything else is either
  - a system program (ships with the operating system), or
  - an application program.
Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EEPROM, generally known as **firmware**
  - Initializes all aspects of system
    - CPU registers, device controllers, memory contents
  - Loads operating system kernel and starts execution
Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles. A memory controller synchronizes access to the memory.
I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- Device controller informs CPU that it has finished its operation by causing an interrupt
Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines.

- Interrupt architecture must save the address of the interrupted instruction.

- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request.

- An operating system is **interrupt driven**.
Interrupt Timeline

- CPU: user process executing
- I/O device: idle, transferring

Timeline:
- I/O request
- Transfer done
- I/O request
- Transfer done
CPU can load instructions only from memory.

The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don’t have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture’s native unit of data. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.  
A **kilobyte**, or **KB**, is 1,024 bytes  
a **megabyte**, or **MB**, is 1,024² bytes  
a **gigabyte**, or **GB**, is 1,024³ bytes  
a **terabyte**, or **TB**, is 1,024⁴ bytes  
a **petabyte**, or **PB**, is 1,024⁵ bytes

Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).
Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - Random access
  - Typically volatile
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
- Hard disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into tracks, which are subdivided into sectors
  - The disk controller determines the logical interaction between the device and the computer
- Solid-state disks – faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular
Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility

- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage

- **Device Driver** for each device controller to manage I/O
  - Provides uniform interface between controller and kernel
Storage-Device Hierarchy

- registers
- cache
- main memory
- solid-state disk
- hard disk
- optical disk
- magnetic tapes
Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy
Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte
A von Neumann architecture
Most systems use a single general-purpose processor
- Most systems have special-purpose processors as well, like device controllers

**Multiprocessors** systems growing in use and importance
- Also known as **parallel systems, tightly-coupled systems**
- Advantages include:
  1. Increased throughput
  2. Economy of scale
  3. Increased reliability – graceful degradation or fault tolerance
- Two types:
  1. **Asymmetric Multiprocessing** – each processor is assigned a specific task: boss and worker processors.
  2. **Symmetric Multiprocessing** – each processor performs all tasks
Symmetric Multiprocessing Architecture

- **CPU<sub>0</sub>**
  - registers
  - cache

- **CPU<sub>1</sub>**
  - registers
  - cache

- **CPU<sub>2</sub>**
  - registers
  - cache

- memory
A Dual-Core Design

- Multi-chip and **multicore**
- Multicore system is more efficient than multi-chip system
  - On-chip communication is faster than between-chip communication.
Clustered Systems

- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** has multiple nodes running applications, monitoring each other
  - Some clusters are for **high-performance computing (HPC)**
    - Applications must be written to use **parallelization**
Operating System Structure

- **Multiprogramming (Batch system)** needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job

- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇒ **process**
  - If several jobs ready to run at the same time ⇒ **CPU scheduling**
  - If processes don’t fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory
Memory Layout for Multiprogrammed System

```
0
operating system

job 1

job 2

job 3

job 4

512M
```
Interrupt driven (hardware and software)

- Hardware interrupt by one of the devices
- Software interrupt (exception or trap):
  - Software error (e.g., division by zero)
  - Request for operating system service
  - Other process problems include infinite loop, processes modifying each other or the operating system
Dual-mode operation allows OS to protect itself and other system components

- **User mode** and **kernel mode**
- **Mode bit** provided by hardware (e.g., CS register in CPU)
  - Provides ability to distinguish when system is running user code or kernel code
  - Some instructions designated as privileged, only executable in kernel mode
  - System call changes mode to kernel, return from call resets it to user
A process is a program in execution. It is a unit of work within the system. Program is a **passive entity**, process is an **active entity**.

Process needs resources to accomplish its task
- CPU, memory, I/O, files
- Initialization data

Process termination requires reclaim of any reusable resources

Single-threaded process has one **program counter** specifying location of next instruction to execute
- Process executes instructions sequentially, one at a time, until completion

Multi-threaded process has one program counter per thread

Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
- Concurrency by multiplexing the CPUs among the processes / threads
Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling
To execute a program all (or part) of the instructions must be in memory.

All (or part) of the data that is needed by the program must be in memory.

Memory management determines what is in memory and when

- Optimizing CPU utilization and computer response to users

Memory management activities

- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes (or parts thereof) and data to move into and out of memory
- Allocating and deallocating memory space as needed
File-system Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - *file*
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Backup files onto stable (non-volatile) storage media
Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write)
### Performance of Various Levels of Storage

<table>
<thead>
<tr>
<th>Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>registers</td>
<td>cache</td>
<td>main memory</td>
<td>solid state disk</td>
<td>magnetic disk</td>
</tr>
<tr>
<td>Typical size</td>
<td>&lt; 1 KB</td>
<td>&lt; 16 MB</td>
<td>&lt; 64 GB</td>
<td>&lt; 1 TB</td>
<td>&lt; 10 TB</td>
</tr>
<tr>
<td>Implementation technology</td>
<td>custom memory with multiple ports CMOS</td>
<td>on-chip or off-chip CMOS SRAM</td>
<td>CMOS SRAM</td>
<td>flash memory</td>
<td>magnetic disk</td>
</tr>
<tr>
<td>Access time (ns)</td>
<td>0.25 - 0.5</td>
<td>0.5 - 25</td>
<td>80 - 250</td>
<td>25,000 - 50,000</td>
<td>5,000,000</td>
</tr>
<tr>
<td>Bandwidth (MB/sec)</td>
<td>20,000 - 100,000</td>
<td>5,000 - 10,000</td>
<td>1,000 - 5,000</td>
<td>500</td>
<td>20 - 150</td>
</tr>
<tr>
<td>Managed by</td>
<td>compiler</td>
<td>hardware</td>
<td>operating system</td>
<td>operating system</td>
<td>operating system</td>
</tr>
<tr>
<td>Backed by</td>
<td>cache</td>
<td>main memory</td>
<td>disk</td>
<td>disk</td>
<td>disk or tape</td>
</tr>
</tbody>
</table>

- Movement between levels of storage hierarchy can be explicit or implicit
  - Hardware function: cache to CPU and registers
  - OS function: disk to memory
Migration of data “A” from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy.

- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache.

- Distributed environment situation even more complex:
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 17
I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering, caching, spooling
  - General device-driver interface
  - Drivers for specific hardware devices
Computing Environments - Traditional

- Stand-alone general purpose machines
- But the line separating traditional computing environments have blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks
Computing Environments - Mobile

- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like augmented reality
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are Apple iOS and Google Android
Computing Environments – Distributed

- Distributed computing
  - Collection of separate, possibly heterogeneous, systems networked together
    - Network is a communications path, TCP/IP most common
      - Local Area Network (LAN)
      - Wide Area Network (WAN)
      - Metropolitan Area Network (MAN)
      - Personal Area Network (PAN)
  - Network Operating System provides features between systems across network
    - E.g., file sharing and communication scheme to allow systems to exchange messages
    - Illusion of a single system
Computing Environments – Client-Server

- A specialized distributed system
- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now servers, responding to requests generated by clients
    - Compute-server system provides an interface to client to request services (i.e., database)
    - File-server system provides interface for clients to store and retrieve files
Computing Environments - Peer-to-Peer

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via discovery protocol
- Examples include Napster and Gnutella, Voice over IP (VoIP) such as Skype
Computing Environments - Virtualization

- Allows operating systems to run as applications within other OSes
  - Vast and growing industry
- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers
Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for its functionality.
  - Amazon EC2 has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company’s own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)
Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools

- Internet connectivity requires security like firewalls
- Load balancers spread traffic across multiple applications
Real-time embedded systems most prevalent form of computers

- Vary considerable, special purpose, limited purpose OS, 
  real-time OS
- Use expanding

Many other special computing environments as well

- Some have OSes, some perform tasks without an OS

Real-time OS has well-defined fixed time constraints

- Processing must be done within constraint
- Correct operation only if constraints met
Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source
- Counter to the copy protection and Digital Rights Management (DRM) movement
- Started by Free Software Foundation (FSF), which has “copyleft” GNU Public License (GPL)
- Examples include GNU/Linux and BSD UNIX (including core of Mac OS X), and many more
End of Chapter 1