Chapter 1: Introduction
Operating Systems
MSc. Ivan A. Escobar

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.

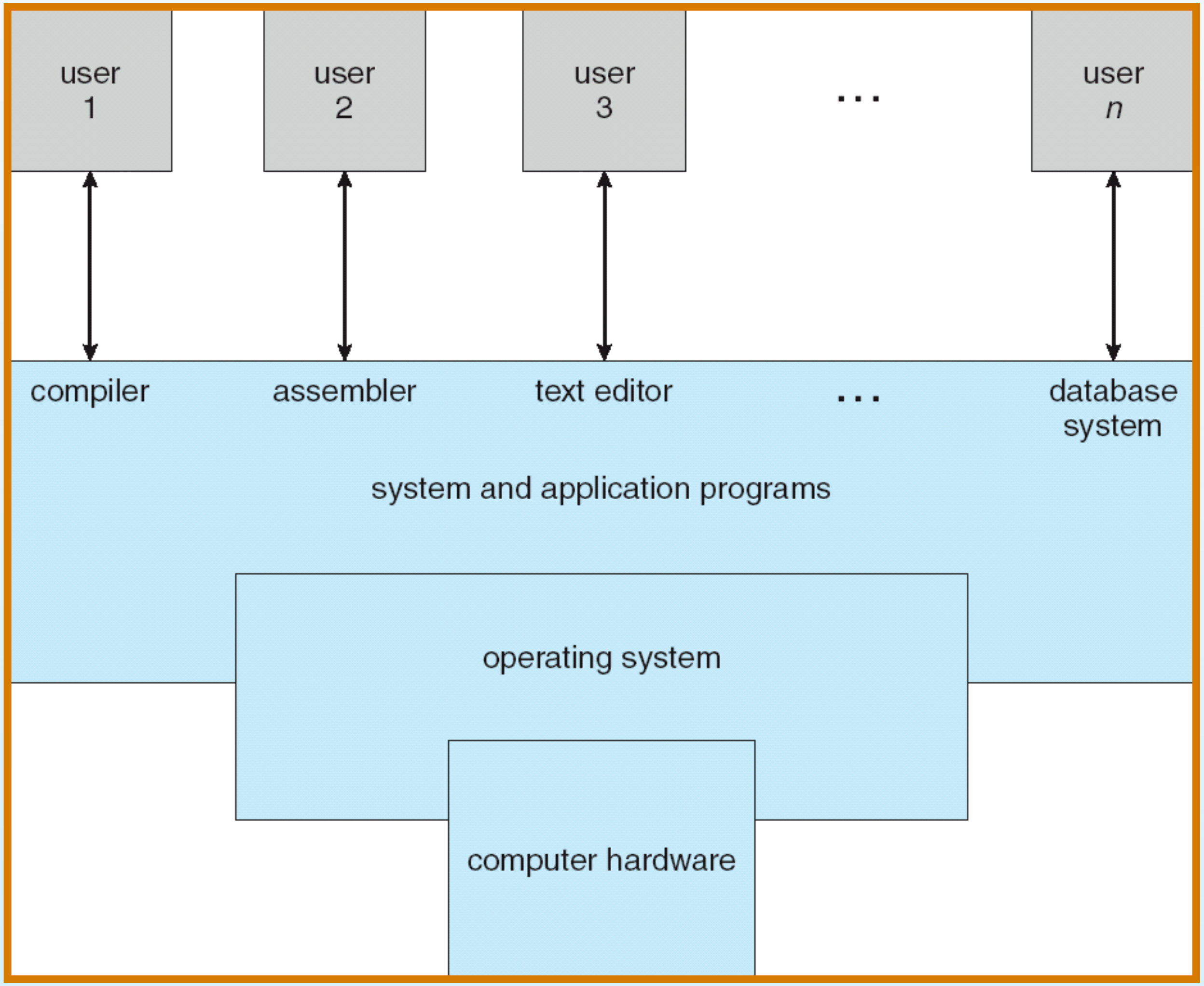
What is an Operating System?

- Separates applications from the hardware they access
  - Software layer
  - Manages software and hardware to produce desired results
- Operating systems primarily are resource managers
  - Hardware
    - Processors
    - Memory
    - Input/output devices
    - Communication devices
  - Software applications

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



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.)

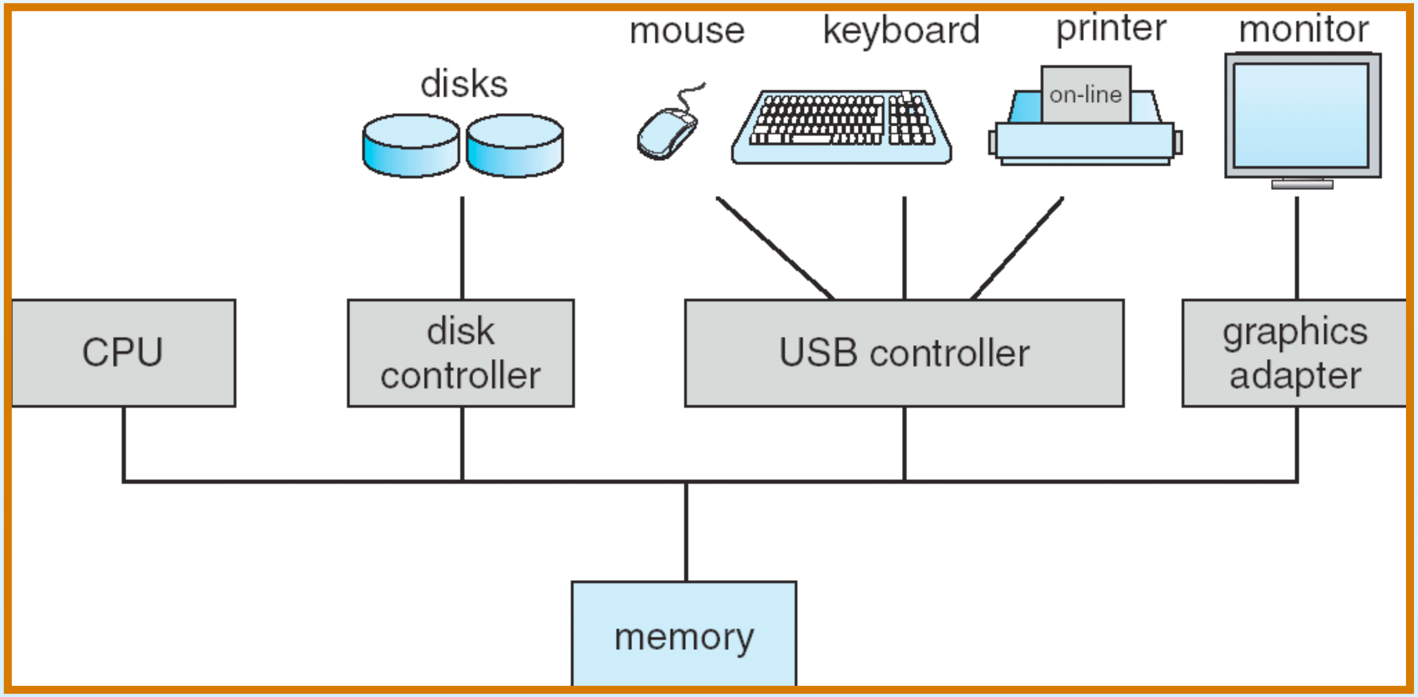
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel.** Everything else is either a system program (ships with the operating system) or an application program

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializates all aspects of system
  - 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



Computer-System Operation

- 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
- I/O is from the device to local buffer of controller.
- 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.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
- An operating system is *interrupt* driven.

Interrupt Handling

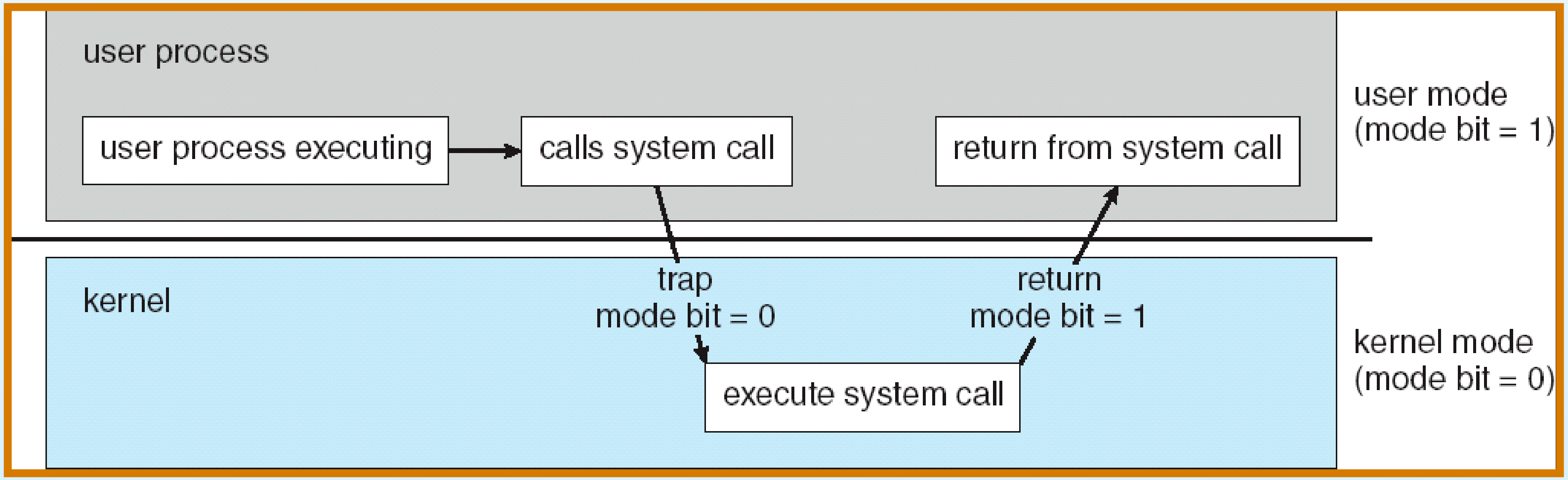
- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - *polling*
  - *vectored* interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

Operating-System Operations

- Interrupt driven by hardware
- Software error or request creates **exception** or **trap**
  - 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
    - 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

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



1.3 Early History: The 1940s and 1950s

- Operating systems evolved through several phases
  - 1940s
    - Early computers did not include operating systems
  - 1950s
    - Executed one job at a time
    - Included technologies to smooth job-to-job transitions
    - Single-stream batch-processing systems
    - Programs and data submitted consecutively on tape

1.4 The 1960s

- 1960s
  - Still batch-processing systems
  - Process multiple jobs at once
    - Multiprogramming
  - One job could use processor while other jobs used peripheral devices
  - Advanced operating systems developed to service multiple interactive users
- 1964
  - IBM announced System/360 family of computers

1.5 The 1970s

- Primarily multimode timesharing systems
  - Supported batch processing, timesharing and real-time applications
  - Personal computing only in incipient stages
    - Fostered by early developments in microprocessor technology
- Department of Defense develops TCP/IP
  - Standard communications protocol
  - Widely used in military and university settings
  - Security problems
    - Growing volumes of information passed over vulnerable communications lines.

1.6 The 1980s

- 1980s
  - Decade of personal computers and workstations
  - Computing distributed to sites at which it was needed
  - Personal computers proved relatively easy to learn and use
    - Graphical user interfaces (GUI)
  - Transferring information between computers via networks became more economical and practical

1.6 The 1980s

- Client/server computing model became widespread
  - Clients request various services
  - Servers perform requested services
- Software engineering field continued to evolve
  - Major thrust by the United States government aimed at tighter control of Department of Defense software projects
    - Realizing code reusability
    - Greater degree of abstraction in programming languages
    - Multiple threads of instructions that could execute independently

1.7 History of the Internet and World

- Advanced Research Projects Agency (ARPA)
  - Department of Defense
  - In late 1960s, created and implemented ARPAnet
    - Grandparent of today’s Internet
    - Networked main computer systems of ARPA-funded institutions
    - Capable of near-instant communication via e-mail
    - Designed to operate without centralized control

1.7 History of the Internet and World

- Transmission Control Protocol/Internet Protocol
  - Set of rules for communicating over ARPANet
  - TCP/IP manages communication between applications
  - Ensure that messages routed properly from sender to receiver
    - Error-correction
  - Later opened to general commercial use

1.7 History of the Internet and World

- World Wide Web (WWW)
  - Locate and view multimedia-based documents on almost any subject
  - Early development begun in 1989 at CERN by Tim Berners-Lee
  - Technology for sharing information via hyperlinked text documents
  - HyperText Markup Language (HTML)
    - Defines documents on WWW
  - Hypertext Transfer Protocol (HTTP)
    - Communications backbone used to transfer documents across WWW

1.8 The 1990s

- Hardware performance improved exponentially
  - Inexpensive processing power and storage
    - Execute large, complex programs on personal computers.
    - Economical machines for extensive database and processing jobs
    - Mainframes rarely necessary
  - Shift toward distributed computing rapidly accelerated
    - Multiple independent computers performing common task

1.8 The 1990s

- Operating system support for networking tasks became standard
  - Increased productivity and communication
- Microsoft Corporation became dominant
  - Windows operating systems
    - Employed many concepts used in early Macintosh operating systems
    - Enabled users to navigate multiple concurrent applications with ease.
- Object technology became popular in many areas of computing
  - Many applications written in object-oriented programming languages
    - For example, C++ or Java
  - Object-oriented operating systems (OOOS)
    - Objects represent components of the operating system
  - Concepts such as inheritance and interfaces
    - Exploited to create modular operating systems

1.8 The 1990s

- Most commercial software sold as object code
  - The source code not included
  - Enables vendors to hide proprietary information and programming techniques
- Free and open-source software became increasingly common in the 1990s
  - Open-source software distributed with the source code
    - Allows individuals to examine and modify software
    - Linux operating system and Apache Web server both open-source
- Richard Stallman launched the GNU project
  - Recreate and extend tools for AT&T’s UNIX operating system

1.8 The 1990s

- Open Source Initiative (OSI)
  - Founded to further benefits of open-source programming
  - Facilitates enhancements to software products
    - Permits anyone to test, debug and enhance applications
  - Increases chance that subtle bugs will be caught and fixed
    - Crucial for security errors which need to be fixed quickly
  - Individuals and corporations can modify the source
    - Create custom software to meet needs of certain environment

1.8 The 1990s

- Operating systems became increasingly user friendly
  - GUI features pioneered by Apple widely used and improved
  - “Plug-and-play” capabilities built into operating systems
    - Enable users to add and remove hardware components dynamically
    - No need to manually reconfigure operating system

1.9 2000 and Beyond

- Middleware
  - Links two separate applications
    - Often over a network and between incompatible machines
  - Particularly important for Web services
    - Simplifies communication across multiple architectures
- Web services
  - Encompass set of related standards
  - Ready-to-use pieces of software on the Internet
  - Enable any two applications to communicate and exchange data

1.11 Operating System Environments

- Operating systems intended for high-end environments
  - Special design requirements and hardware support needs
    - Large main memory
    - Special-purpose hardware
    - Large numbers of processes
- Embedded systems
  - Characterized by small set of specialized resources
  - Provide functionality to devices such as cell phones and PDAs
  - Efficient resource management key to building successful operating system

1.11 Operating System Environments

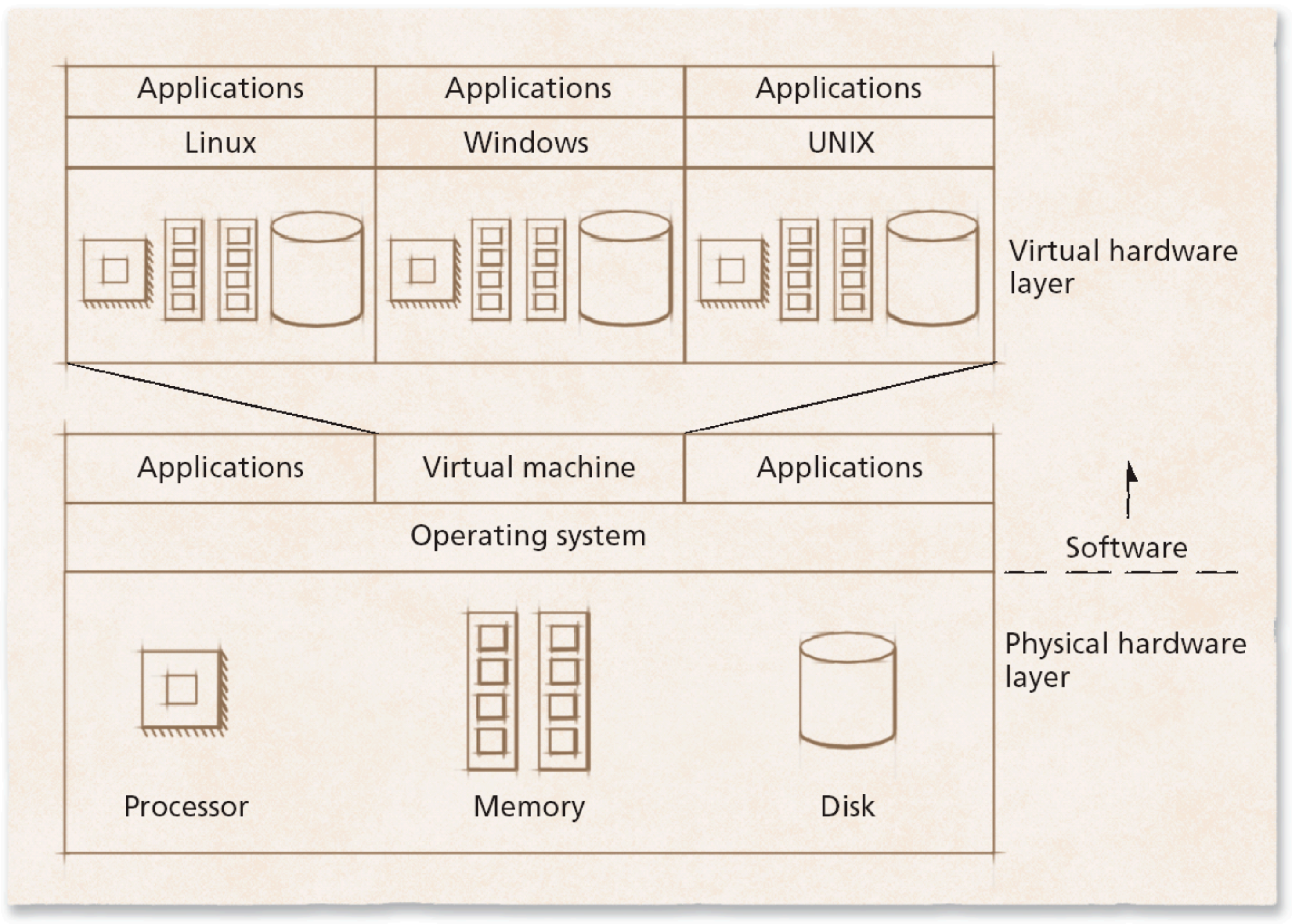
- Real-time systems
  - Require that tasks be performed within particular (often short) time frame
    - Autopilot feature of an aircraft must constantly adjust speed, altitude and direction
  - Such actions cannot wait indefinitely—and sometimes cannot wait at all

1.11 Operating System Environments

- Virtual machines (VMs)
  - Software abstraction of a computer
  - Often executes on top of native operating system
- Virtual machine operating system
  - Manages resources provided by virtual machine
- Applications of virtual machines
  - Allow multiple instances of an operating system to execute concurrently
  - Emulation
    - Software or hardware mimics functionality of hardware or software not present in system
  - Promote portability

1.11 Operating System Environments

Figure 1.2 Schematic of a virtual machine.



1.12 Operating System Components and

- Computer systems have evolved
  - Early systems contained no operating system,
  - Later gained multiprogramming and timesharing machines
  - Personal computers and finally truly distributed systems
  - Filled new roles as demand changed and grew

1.12.1 Core Operating System

- User interaction with operating system
  - Often, through special application called a shell
  - Kernel
    - Software that contains core components of operating system
- Typical operating system components include:
  - Processor scheduler
  - Memory manager
  - I/O manager
  - Interprocess communication (IPC) manager
  - File system manager

1.12.1 Core Operating System

- Multiprogrammed environments now common
  - Kernel manages the execution of processes
  - Program components which execute independently but use single memory space to share data are called threads.
  - To access I/O device, process must issue system call
    - Handled by device driver
    - Software component that interacts directly with hardware
    - Often contains device-specific commands

1.12.2 Operating System Goals

- Users expect certain properties of operating systems
  - Efficiency
  - Robustness
  - Scalability
  - Extensibility
  - Portability
  - Security
  - Protection
  - Interactivity
  - Usability

1.13 Operating System Architectures

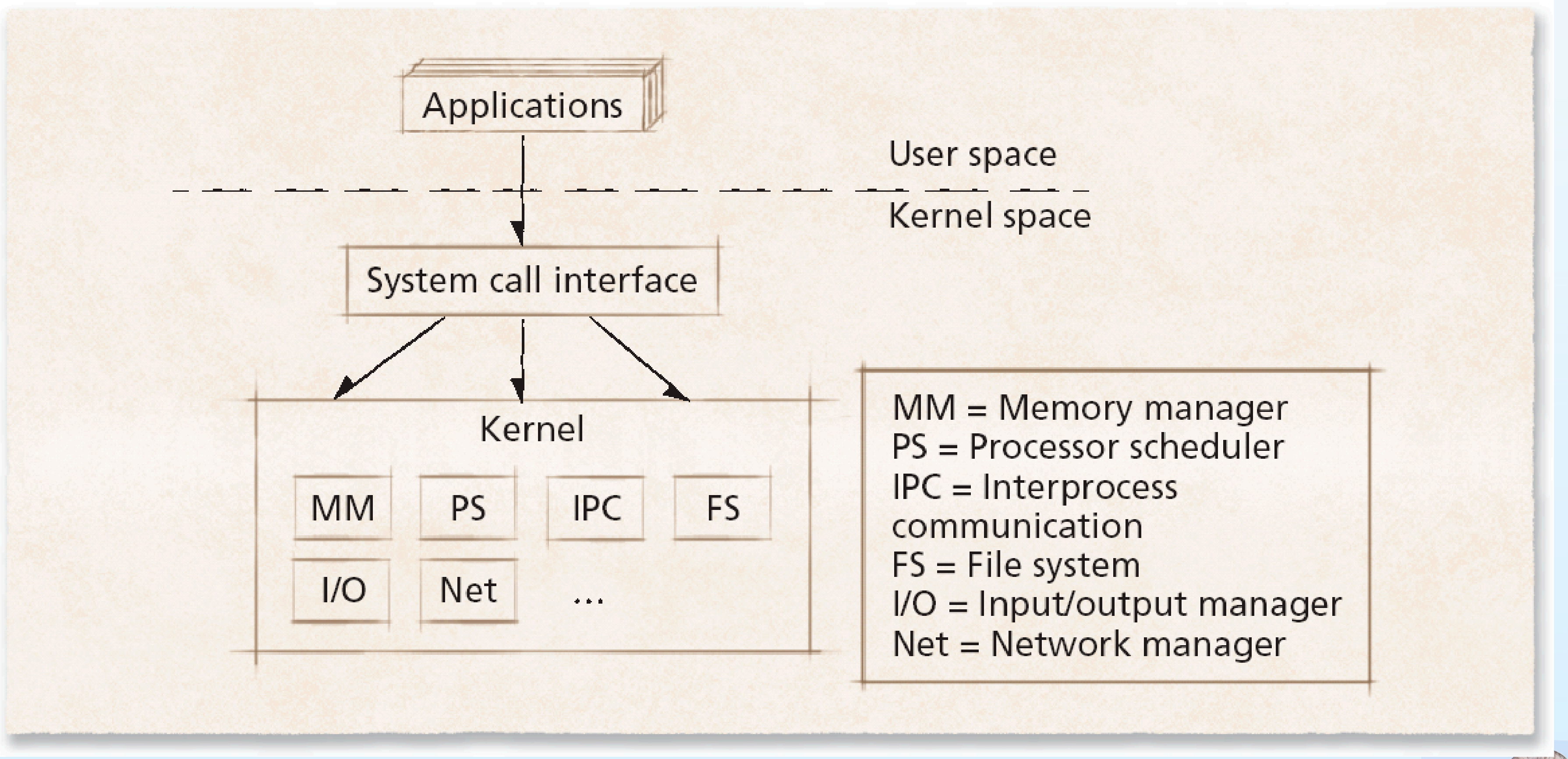
- Today’s operating systems tend to be complex
  - Provide many services
  - Support variety of hardware and software
  - Operating system architectures help manage this complexity
    - Organize operating system components
    - Specify privilege with which each component executes

1.13.1 Monolithic Architecture

- Monolithic operating system
  - Every component contained in kernel
    - Any component can directly communicate with any other
  - Tend to be highly efficient
  - Disadvantage is difficulty determining source of subtle errors

1.13.1 Monolithic Architecture

Figure 1.3 Monolithic operating system kernel architecture.



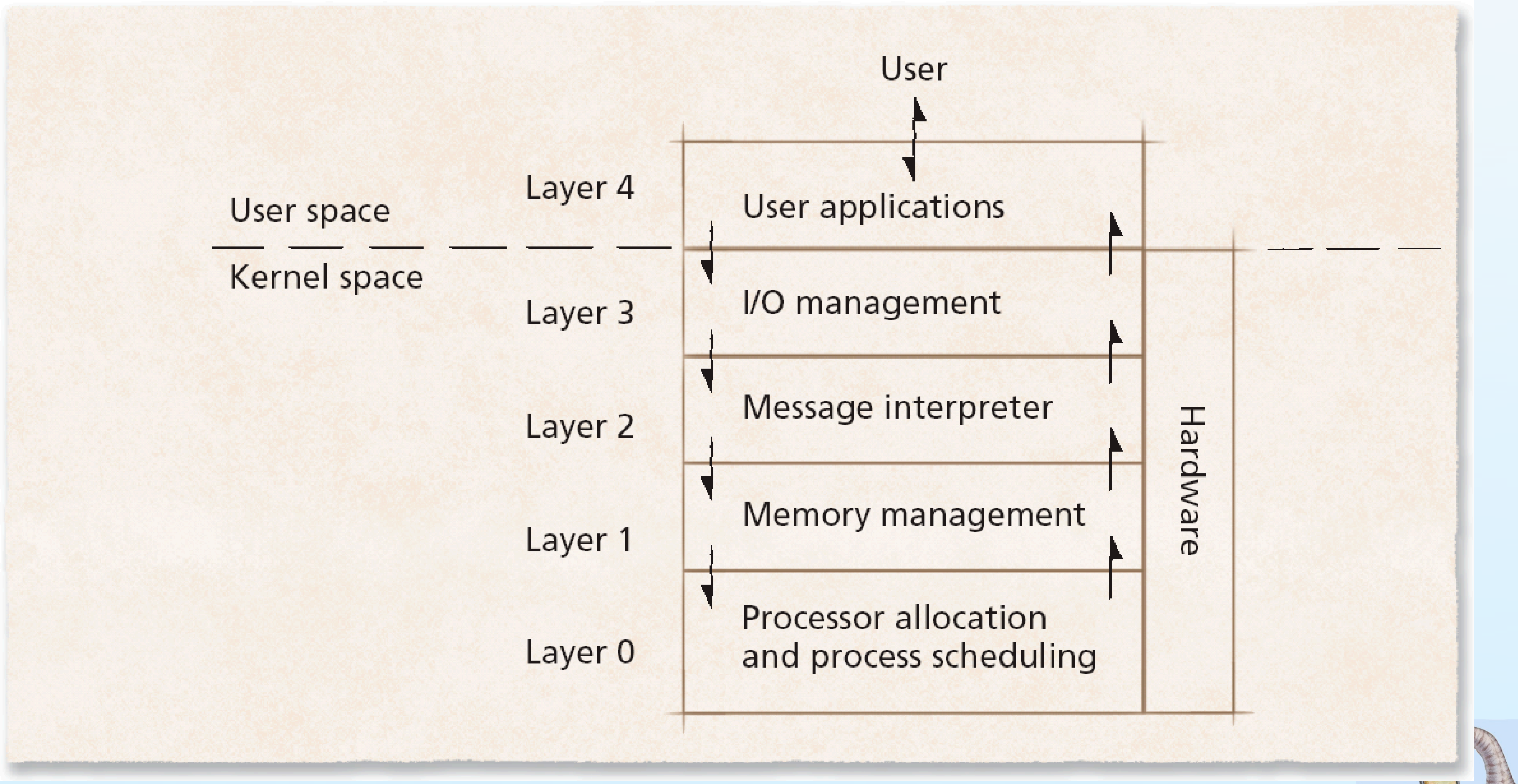
MM = Memory manager
PS = Processor scheduler
IPC = Interprocess communication
FS = File system
I/O = Input/output manager
Net = Network manager

1.13.2 Layered Architecture

- Layered approach to operating systems
  - Tries to improve on monolithic kernel designs
    - Groups components that perform similar functions into layers
  - Each layer communicates only with layers immediately above and below it
  - Processes’ requests might pass through many layers before completion
  - System throughput can be less than monolithic kernels
    - Additional methods must be invoked to pass data and control

1.13.2 Layered Architecture

Figure 1.4 Layers of the THE operating system.

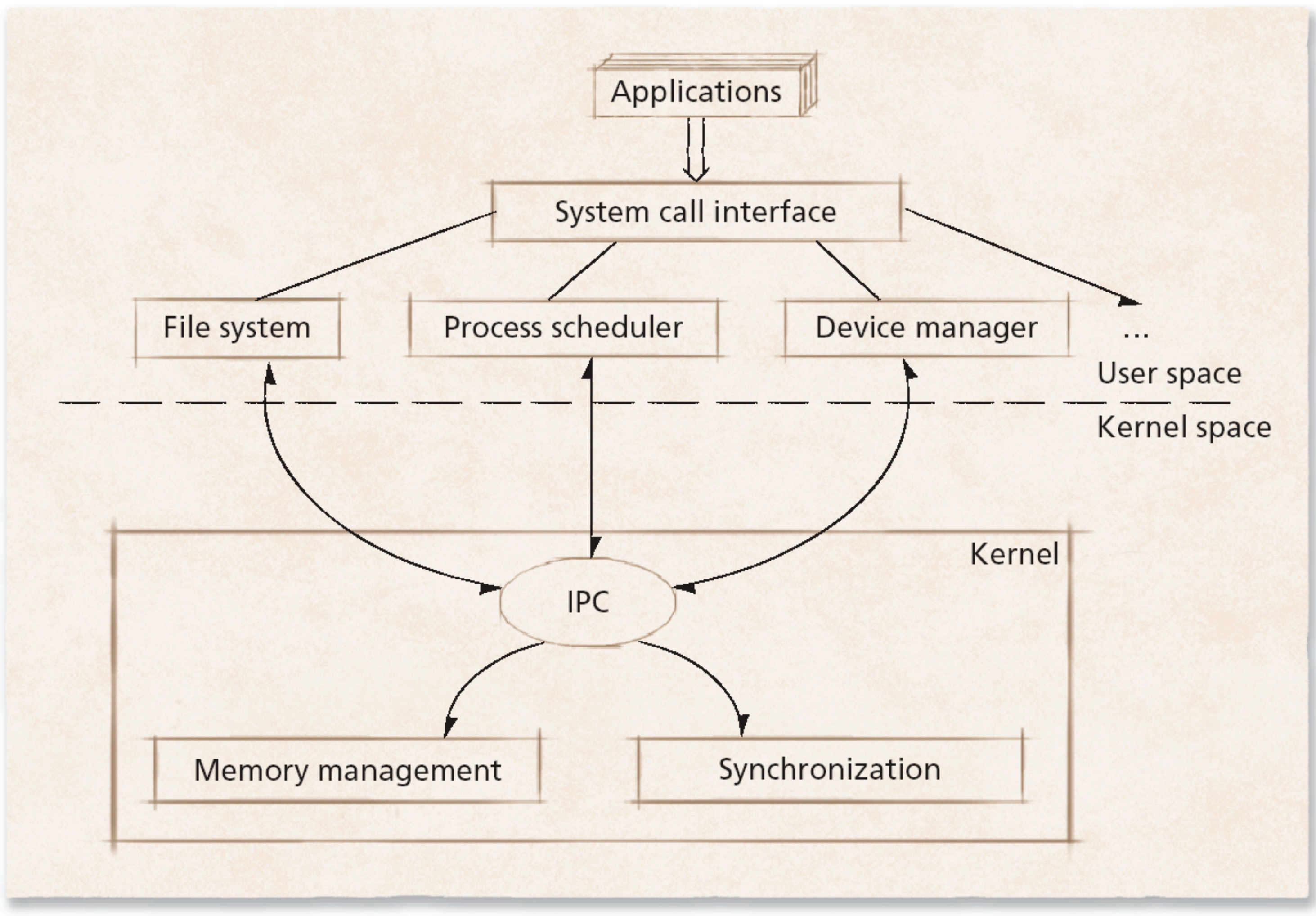


1.13.3 Microkernel Architecture

- Microkernel operating system architecture
  - Provides only small number of services
    - Attempt to keep kernel small and scalable
  - High degree of modularity
    - Extensible, portable and scalable
  - Increased level of intermodule communication
    - Can degrade system performance

1.13.3 Microkernel Architecture

Figure 1.5 Microkernel operating system architecture.

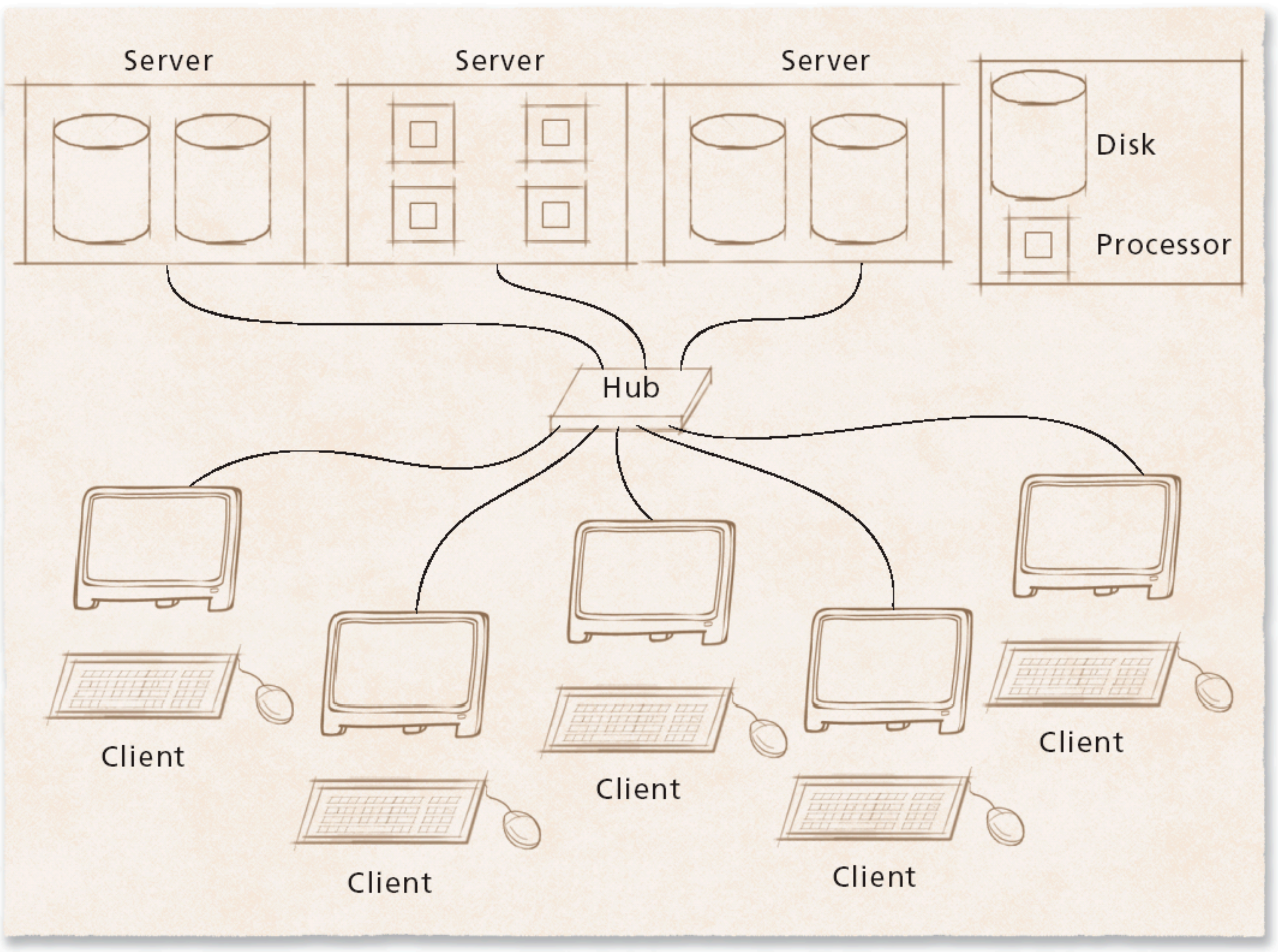


1.13.4 Networked and Distributed Operating Systems

- Network operating system
  - Runs on one computer
  - Allows its processes to access resources on remote computers
- Distributed operating system
  - Single operating system
  - Manages resources on more than one computer system
  - Goals include:
    - Transparent performance
    - Scalability
    - Fault tolerance
    - Consistency

1.13.4 Networked and Distributed Operating Systems

Figure 1.6 Client/server networked operating system model.



End of Chapter 1