COP 4600
Introduction To Operating Systems
Summer 2013

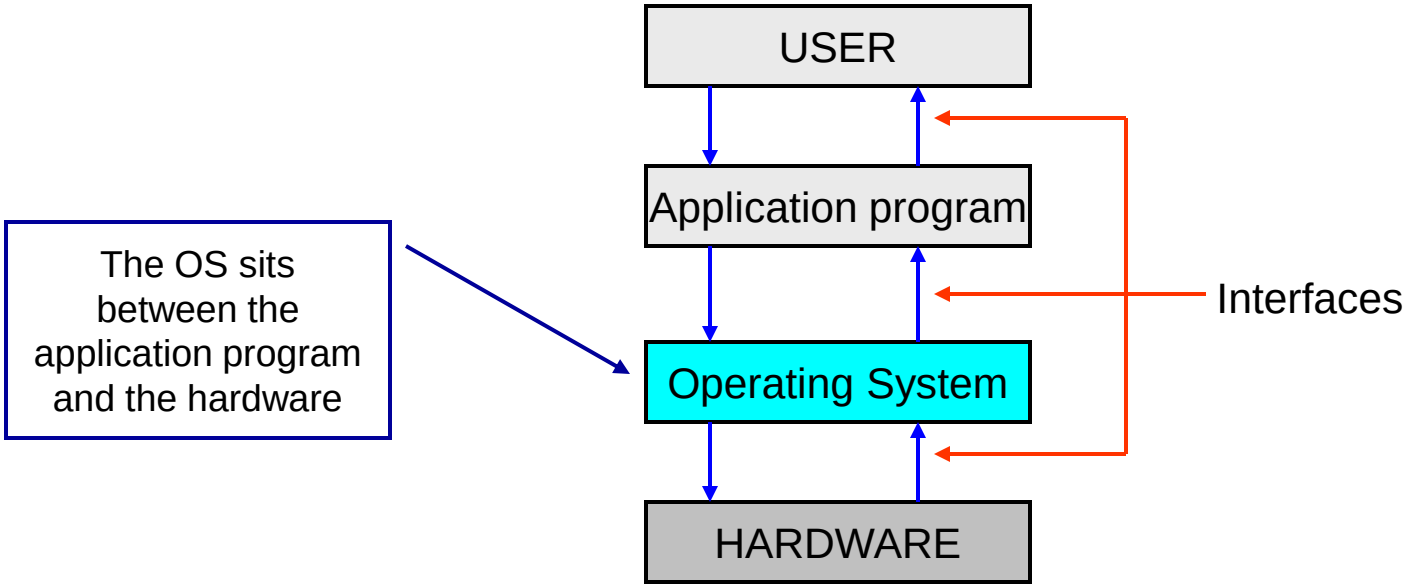
Introductory Material

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop4600/sum2013

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

What is an Operating System?

- In the most general sense an operating system is a collection of system software routines that sit between an application program and the computer hardware on which that application is to be executed.



What Is An Operating System?

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly, some OS require less than 1MB and do not even have a full-screen editor, while others require many GBs and are entirely based on graphical windowing systems.
  - Recall that in 1998 the U.S Department of Justice filed suit against Microsoft, in essence claiming that Microsoft included too much functionality in its OS and thus prevented vendors from competing.
- "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

What Is An Operating System?

- For now, we can think of an OS as:

  1) the interface or intermediary between a user/application and the computer hardware
  2) providing an environment in which the user can execute programs conveniently and
     - application and/or system software
  3) managing the computer's resources efficiently
     - memory, disk space, CPU time, I/O, software, etc.

- Often an OS is a tradeoff between convenience and efficiency.
  – Windows (GUI) vs. Unix (command interpreter)

What Is An Operating System?

- 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

What Is An Operating System?

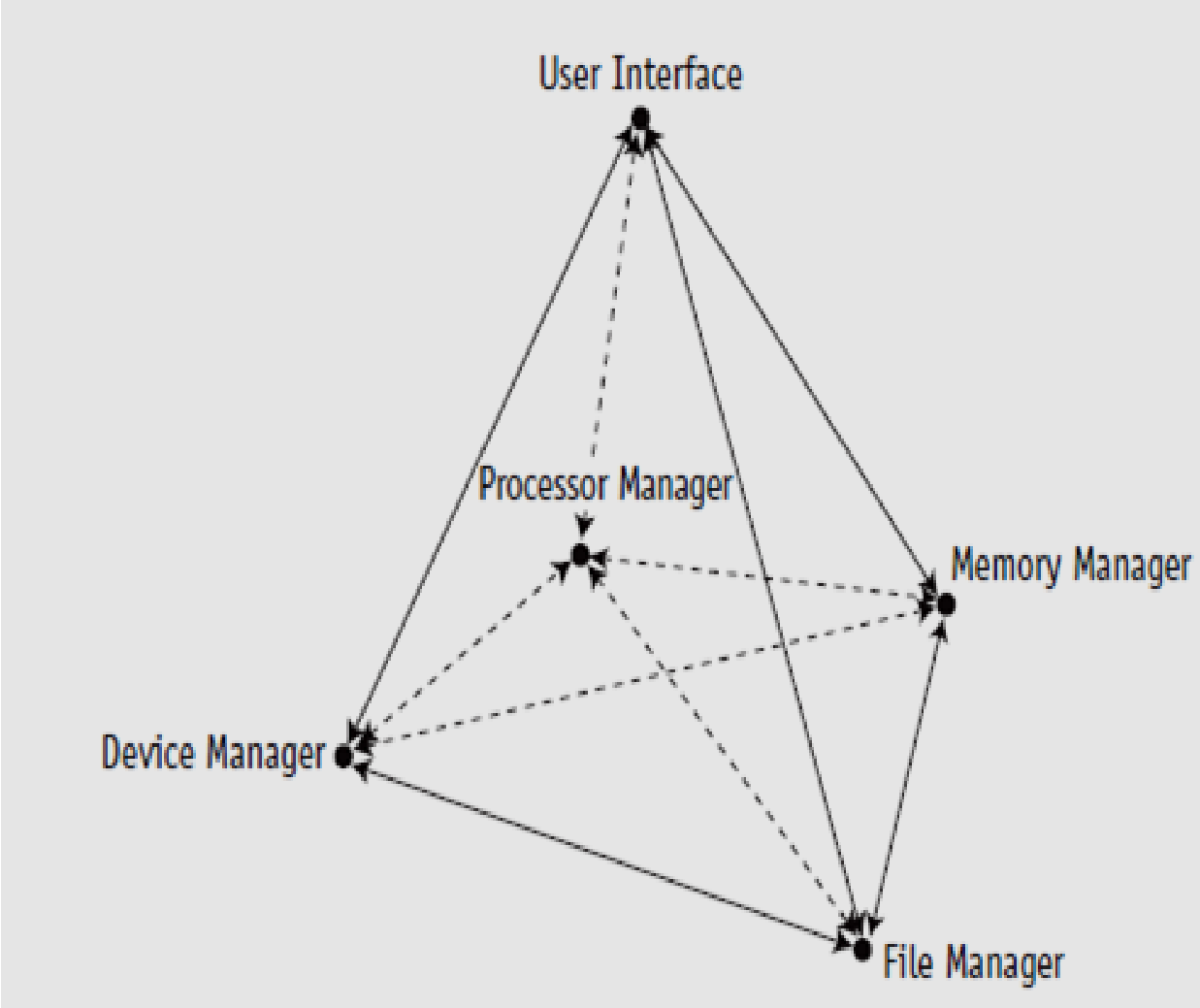
- The diagram on page 8 illustrates an abstract representation of an OS and the interrelationships of its major components.
- The base of the pyramid represents the four essential managers of every OS, the memory manager, the processor manager, the device manager, and the file manager.
- Each manager works closely with the other managers and performs a unique role.
- The User Command Interface is unique to each OS.

What Is An Operating System?

- Each manager:
  - Works closely with other managers
  - Performs a unique role
- Manager tasks include:
  - Monitor its resources continuously
  - Enforce policies determining:
    - Who gets what, when, and how much
  - Allocate the resource (when appropriate)
  - Deallocate the resource (when appropriate)

What Is An Operating System?

This model of a non-networked OS illustrates the four sub-system managers supporting the User Interface.

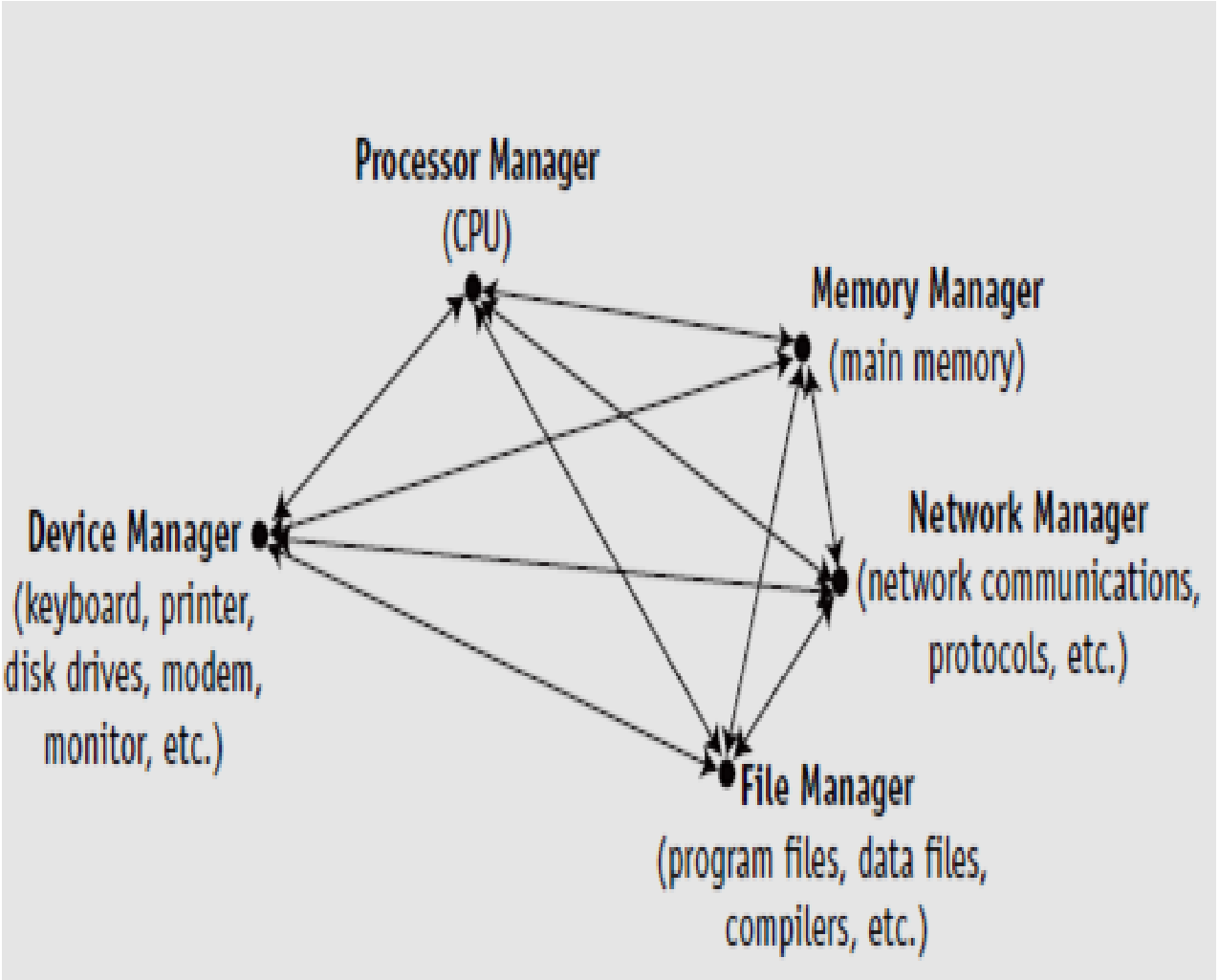


What Is An Operating System?

- Virtually every modern OS includes networking capabilities.
- This adds a fifth essential manager called the network manager that provides a mechanism for users to share resources while controlling users' access to them.
- The diagram on page 10 illustrates a modified abstract representation of an OS and the interrelationships of the essential managers assuming networked capabilities using a five-sided pyramid.

What Is An Operating System?

Networked systems have a Network Manager that assumes responsibility for networking tasks while working harmoniously with every other manager.



Main Memory Management

- In charge of main memory
  - Random Access Memory (RAM)
- Responsibilities include:
  - Preserving space in main memory occupied by operating system
  - Checking validity and legality of memory space request
  - Setting up memory tracking table
    - Tracks usage of memory by sections
    - Needed in multiuser environment
  - Deallocating memory to reclaim it

Processor Management

- In charge of allocating **Central Processing Unit** (CPU)
- Tracks **process** status
  - An instance of program execution
- Two levels of responsibility:
  - Handle jobs as they enter the system
    - Handled by Job Scheduler
  - Manage each process within those jobs
    - Handled by Process Scheduler

Device Management

- In charge of monitoring all resources
  - Devices, channels, and control units
- Responsibilities include:
  - Choosing most efficient resource allocation method
    - Printers, ports, disk drives, etc.
    - Based on scheduling policy
  - Allocating the device
  - Starting device operation
  - Deallocating the device

File Management

- In charge of tracking every file in the system
  - Data files, program files, compilers, application programs
- Responsibilities include:
  - Enforcing user/program resource access restrictions
    - Uses predetermined access policies
  - Controlling user/program modification restrictions
    - Read-only, read-write, create, delete
  - Allocating resource
    - Opening the file
    - Deallocating file (by closing it)

A Brief History Of Machine Hardware

(apologies to Steven Hawking for the title)

- **Hardware:** physical machine and electronic components
  - **Main memory (RAM)**
    - Data/Instruction storage and execution
  - **Input/Output devices (I/O devices)**
    - All peripheral devices in system
    - Printers, disk drives, CD/DVD drives, flash memory, and keyboards
  - **Central processing unit (CPU)**
    - Controls interpretation and execution of instructions
    - Controls operation of computer system

A Brief History Of Machine Hardware

- Computer classification
  - By capacity and price (until mid-1970s)
- Mainframe
  - Large machine
    - Physical size and internal memory capacity
  - Classic Example: 1964 IBM 360 model 30
    - CPU required 18-square-foot air-conditioned room
    - CPU size: 5 feet high x 6 feet wide
    - Internal memory: 64K
    - Price: $200,000 (1964 dollars)
  - Applications limited to large computer centers

A Brief History Of Machine Hardware

- **Minicomputer**
  - Developed for smaller institutions
  - Compared to mainframe
  - Smaller in size and memory capacity
    - Cheaper
  - Example: Digital Equipment Corp. minicomputer
    - Price: less than $18,000
  - Today
    - Known as **midrange computers**
    - Capacity between microcomputers and mainframes

A Brief History Of Machine Hardware

- **Supercomputer**
  - Massive machine
  - Developed for military operations and weather forecasting
  - Example: Japan's "K Computer"
    - 705,204 cores
    - Performs up to 10.51 Petaflop/s (10.51×10^{15} flops)
  - Uses:
    - Scientific research
    - Customer support/product development

A Brief History Of Machine Hardware

- **Microcomputer**
  - Developed for single users in the late 1970s
  - Example: microcomputers by Tandy Corporation and Apple Computer, Inc.
    - Very little memory (by today's standards)
    - 64K maximum capacity
  - Microcomputer's distinguishing characteristic
    - Single-user status

A Brief History Of Machine Hardware

- **Workstations**
  - Most powerful microcomputers
  - Developed for commercial, educational, and government enterprises
  - Networked together
  - Support engineering and technical users
    - Massive mathematical computations
    - Computer-aided design (CAD)
  - Applications
    - Requiring powerful CPUs, large main memory, and extremely high-resolution graphic displays

A Brief History Of Machine Hardware

- **Servers**
  - Provide specialized services
    - To other computers or client/server networks
  - Perform critical network task
  - Examples:
    - Print servers
    - Internet servers
    - Mail servers

A Brief History Of Machine Hardware

- **Advances in computer technology**
  - Dramatic changes
    - Physical size, cost, and memory capacity
  - Networking
    - Integral part of modern computer systems
  - Mobile society information delivery
    - Creating strong market for handheld devices
  - New classification
    - By processor capacity, not memory capacity
  - Moore's Law
    - Computing power rises exponentially

Types Of Operating Systems

- Five categories
  - Batch
  - Interactive
  - Real-time
  - Hybrid
  - Embedded
- Two distinguishing features
  - Response time
  - How data enters into the system

Types Of Operating Systems

- **Batch Systems**
  - Input relied on punched cards or tape
  - Efficiency measured in throughput
- **Interactive Systems**
  - Faster turnaround than batch systems
  - Slower than real-time systems
  - Introduced to provide fast turnaround when debugging programs
  - Time-sharing software developed for operating system

Types Of Operating Systems

- **Real-time systems**
  - Reliability is key
  - Fast and time limit sensitive
  - Used in time-critical environments
    - Space flights, airport traffic control, high-speed aircraft
    - Industrial processes
    - Sophisticated medical equipment
    - Distribution of electricity
    - Telephone switching
  - Must be 100% responsive, 100% of the time

Types Of Operating Systems

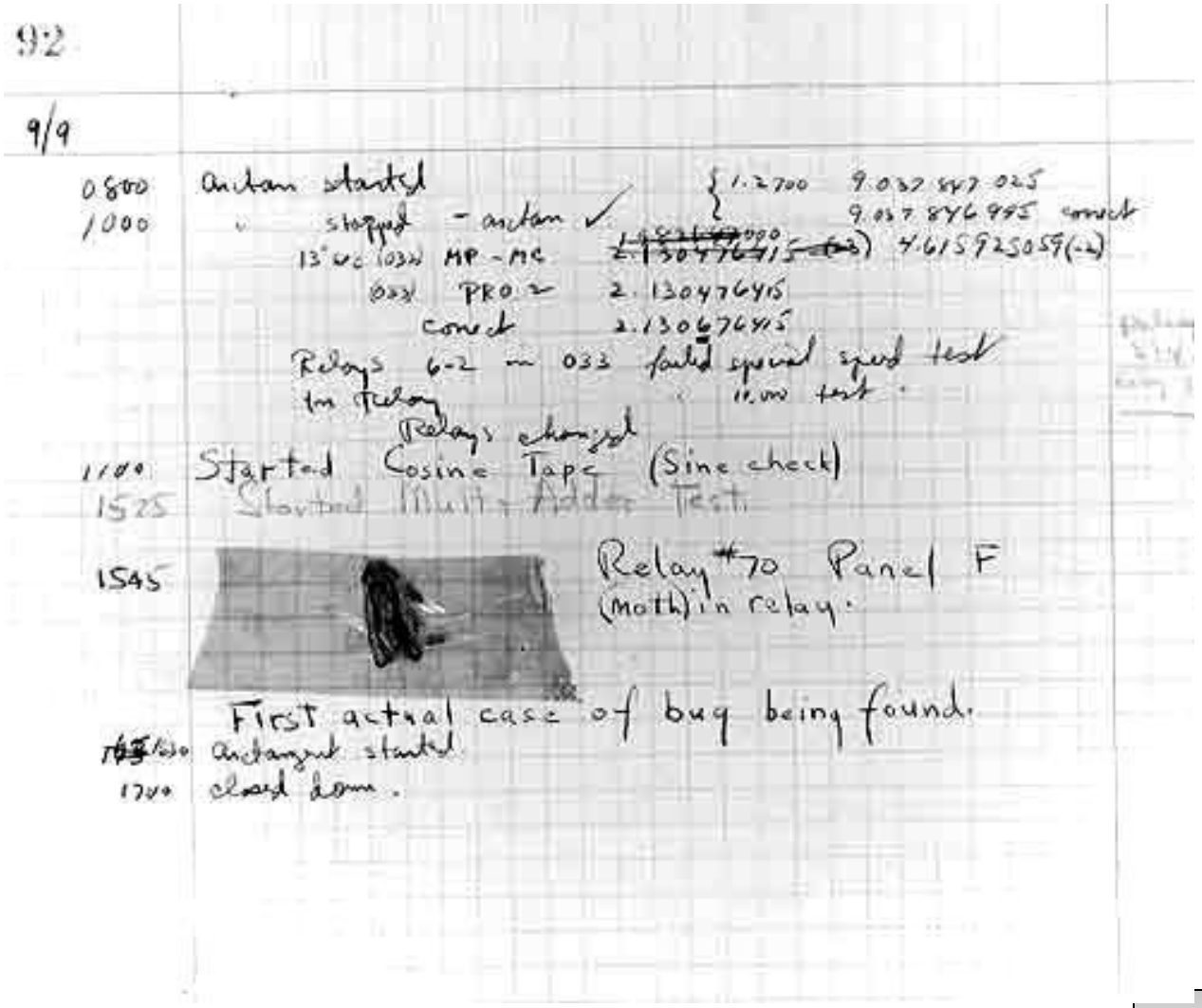
- **Hybrid systems**
  - Combination of batch and interactive
  - Accept and run batch programs in the background
    - Interactive load is light
- **Embedded systems**
  - Computers placed inside other products
  - Adds features and capabilities
  - Operating system requirements
    - Perform specific set of programs
    - Not interchangeable among systems
    - Small kernel and flexible function capabilities

A Brief History of OS Development

- **1940s: first generation**
  - Computers based on vacuum tube technology
  - No standard operating system software
  - Typical program included every instruction needed by the computer to perform the tasks requested
  - Poor machine utilization
    - CPU processed data and performed calculations for fraction of available time
  - Early programs
    - Designed to use the resources conservatively
    - Understandability is not a priority

Types Of Operating Systems

This is a photograph of the research journal maintained by Dr. Grace Hopper from her work on Harvard's Mark 1 computer in 9/9 0800 Arctan started 1000 " stopped - arctan ✓ 1700 closed down. Relay #70 Panel F (moth) in relay. First actual case of bug being found. computer "bug", a moth that was caught in one of the system's relays causing the system to crash. Today's use of the term "bug" stems from that first moth and its resultant system crash.



A Brief History of OS Development

- **1950s: second generation**
  - Focused on cost effectiveness
  - Computers were expensive
    - IBM 7094: $200,000
  - Two widely adopted improvements
    - Computer operators: humans hired to facilitate machine operation
    - Concept of job scheduling: group together programs with similar requirements
  - Expensive time lags between CPU and I/O devices

A Brief History of OS Development

- **1950s: second generation (cont'd.)**
  - I/O device speed gradually became faster
    - Tape drives, disks, and drums
  - Records blocked *before* retrieval or storage
  - Access methods developed
    - Added to object code by linkage editor
  - Buffer between I/O and CPU introduced
    - Reduced speed discrepancy
  - Timer interrupts developed
    - Allowed job-sharing

A Brief History of OS Development

- **1960s: third generation**
  - Faster CPUs
  - Speed caused problems with slower I/O devices
  - Multiprogramming
    - Allowed loading many programs at one time
  - Program scheduling
    - Initiated with second-generation systems
    - Continues today
  - Few advances in data management
  - Total operating system customization
    - Suit user's needs

A Brief History of OS Development

- **1970s**
  - Faster CPUs
  - Speed caused problems with slower I/O devices
  - Main memory physical capacity limitations
    - Multiprogramming schemes used to increase CPU
    - Virtual memory developed to solve physical limitation
  - Database management software
    - Became a popular tool
  - A number of query systems introduced
  - Programs started using English-like words, modular structures, and standard operations

A Brief History of OS Development

- **1980s**
  - **Cost/performance ratio** improvement of computer components
  - More flexible hardware (firmware)
  - **Multiprocessing**
    - Allowed parallel program execution
  - Evolution of personal computers
  - Evolution of high-speed communications
  - **Distributed processing** and **networked systems** introduced

A Brief History of OS Development

- **1990s**
  - Demand for Internet capability
    - Sparked proliferation of networking capability
    - Increased networking
    - Increased tighter security demands to protect hardware and software
  - Multimedia applications
    - Demanding additional power, flexibility, and device compatibility for most operating systems

A Brief History of OS Development

- **2000s**
  - Primary design features support:
    - Multimedia applications
    - Internet and Web access
    - Client/server computing
  - Computer systems requirements
    - Increased CPU speed
    - High-speed network attachments
    - Increased number and variety of storage devices
  - Virtualization
    - Single server supports different operating systems

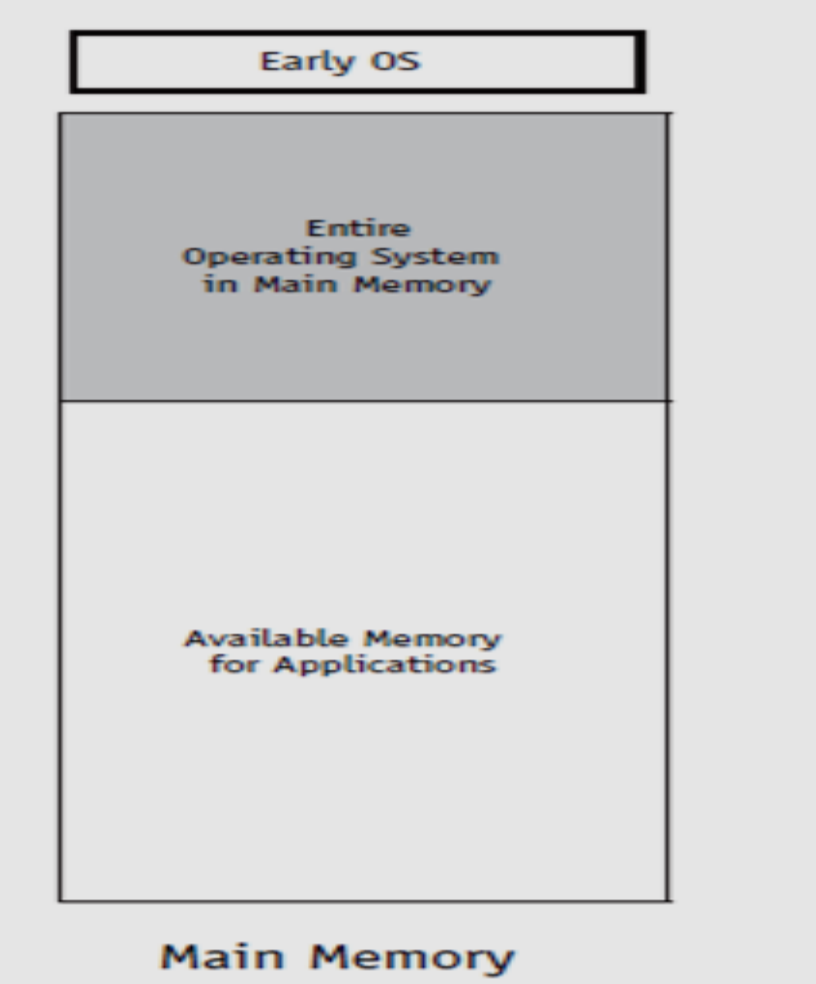
A Brief History of OS Development

- The 1980s marked the end of a clearly delineated evolution of OS design.
- In the 1990s, new design features and new OS were introduced to accommodate new developments in hardware devices and software applications.
- The primary design features of current OS are based on the needs of users, who have come to expect support for multimedia applications, web access, as well as client-server computing. Computer systems help to provide these services with increased CPU speed, high-speed network attachments, and an increased number and variety of storage devices.

A Brief History of OS Development

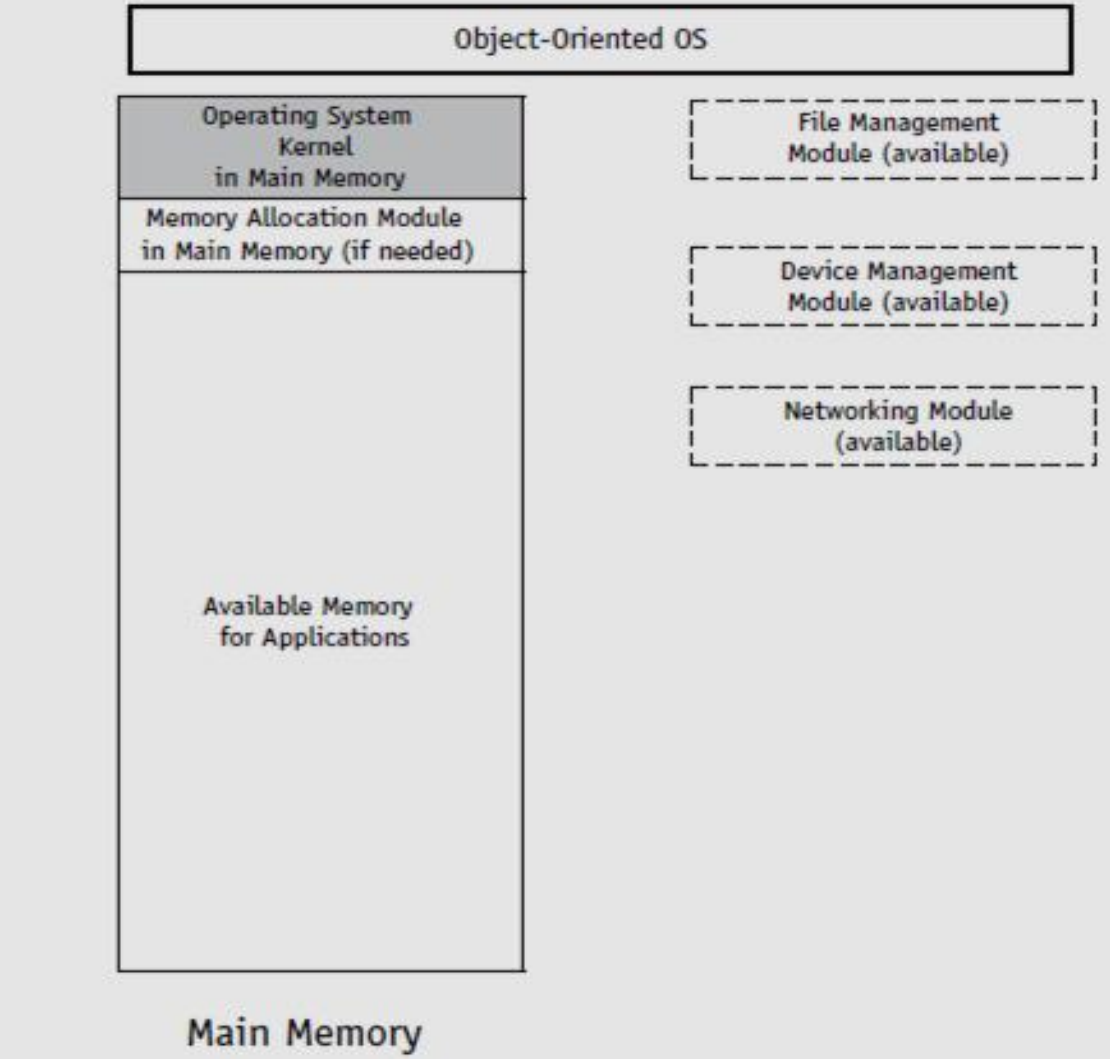
- One are which has made significant improvement in the system architecture of OS – the way their components are programmed and organized, is the use of OO design techniques and the reorganization of the OS nucleus, the **kernel**.
- The kernel is the part of the OS that resides in main memory at all times, performs the most essential OS tasks, and is protected by hardware from user tampering.
- Early OS were designed as shown on the next page, with a very large kernel.
- More recent OS are designed as shown on page 39.

A Brief History of OS Development



A Brief History of OS Development

Modern kernel design limits inclusion to only a few essential functions, such as process scheduling and memory allocation, while all other functions, such as device allocation, are provided by special processes, which are treated as regular applications by the kernel.



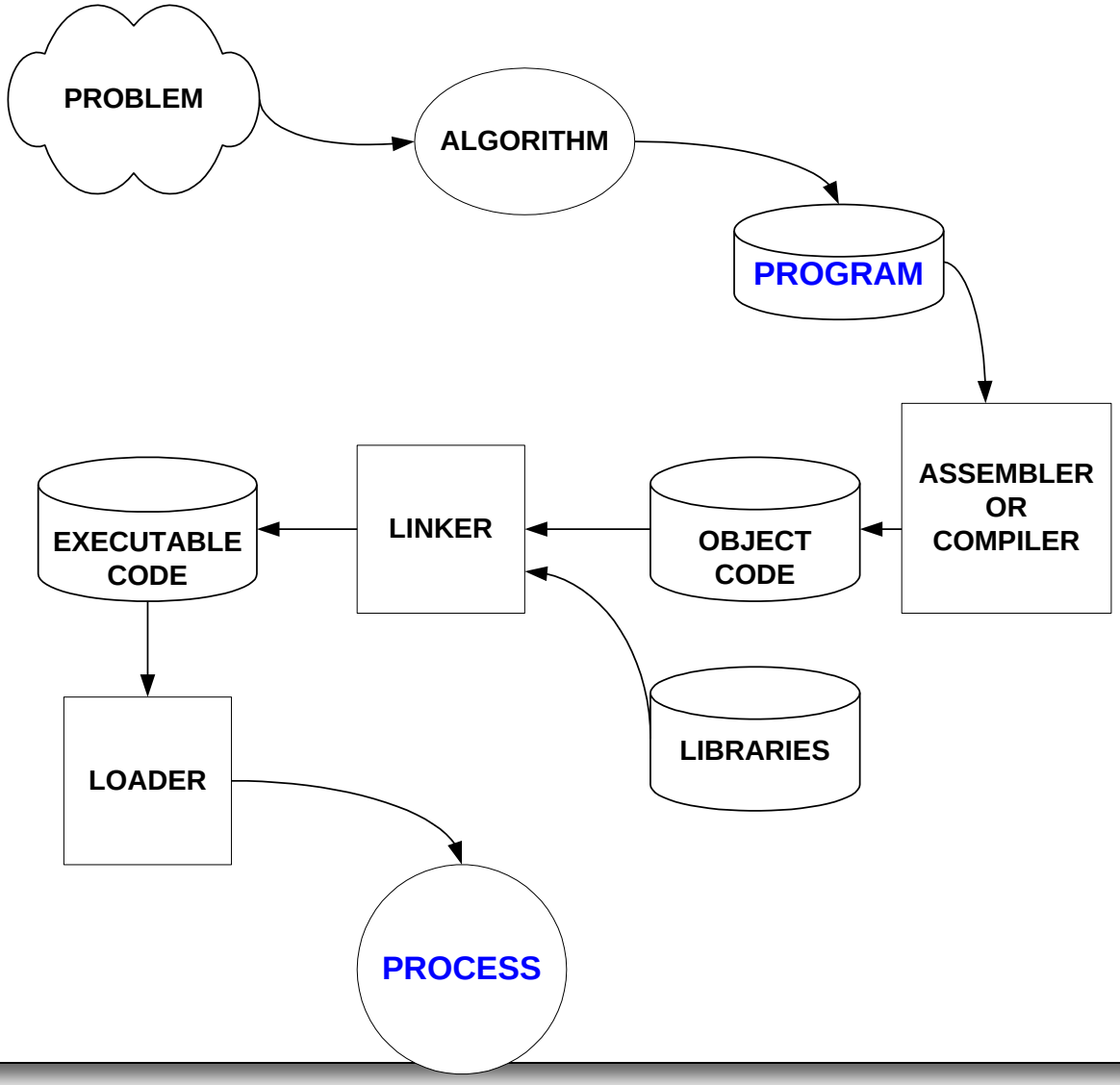
A Brief History of OS Development

- Process handling, which addresses the way in which processes are executed to maximize the system's efficiency and satisfy user needs, is also an area that has benefitted from research and development efforts.
- One of the biggest improvements can be seen in the implementation of threads, multiprocessing, and distributed systems.
- To understand the concept of a thread, you need to understand the concept of a process or task.

A Brief History of OS Development

- A process has two characteristics:
  - It requires space in main memory where it resides during its execution although, from time to time, it requires other resources such as data files or I/O devices.
  - It passes through several states (such as running, waiting, ready, etc.) from its initial arrival into the system until its completion.

What Is A Process?



What Is A Process?

- A *process*:
  - is a program in execution.
  - has a process control block (PCB)
  - has a program counter (PC)

- A process can have one or more *threads*.
  - A thread is sometimes known as a *lightweight process*

What Is A Thread?

- Multiple actions executing simultaneously
  - Heavyweight process (conventional process)
    - Owns the resources
    - Passive element
  - Lightweight process (thread)
    - Uses CPU and scheduled for execution
    - Active element
  - Multithreaded applications programs
    - Contain several threads running at one time
    - Same or different priorities
    - Examples: Web browsers and time-sharing systems

Performance Measurements

- Utilization (maximize)
  - $U = T_{busy} / T_{total}$ where T_{total} is total study time, or
  - $U = T_{used} / T_{available}$

- Throughput (maximize)
  - X = C / T where C is number of completed jobs/processes and T is time frame
  - The rate at which requests are processed

Performance Measurements

- Turnaround Time (minimize)
  - Typically used in reference to batch systems
  - The time it takes to complete/execute a job/process

- Response Time (minimize)
  - Typically used in reference to interactive systems
  - The time it takes for the system to respond to a user request from submission to start of a response