CSE 380
Computer Operating Systems

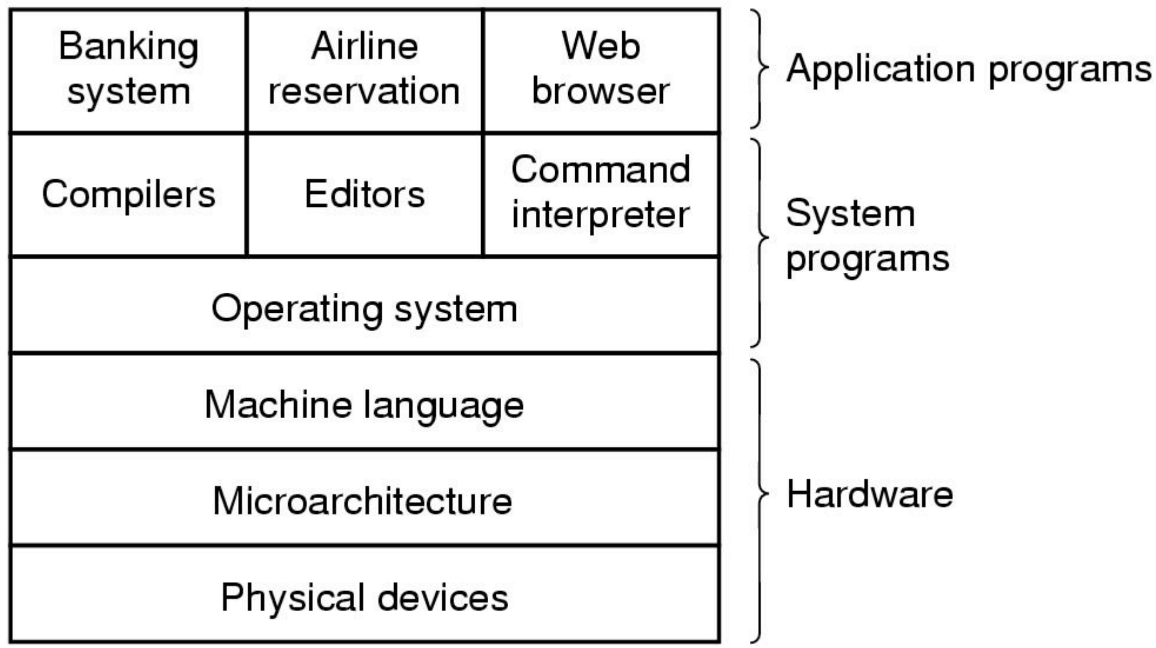
Instructor: Insup Lee

University of Pennsylvania

Fall 2003

Lecture Note 1: Introduction

What is an Operating System?

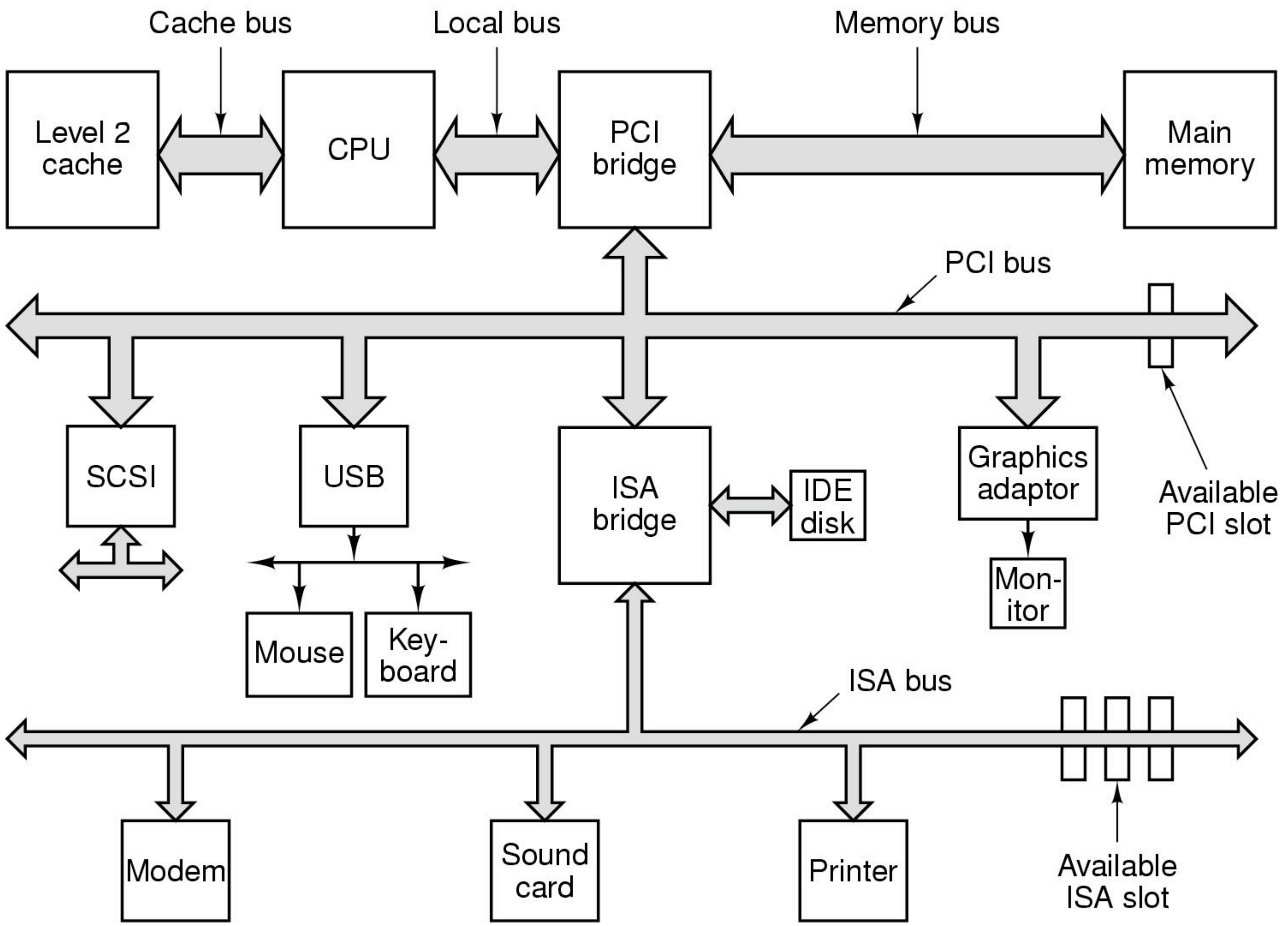


Operating systems provides an interface between hardware and user programs, and makes hardware usable

Resource Abstraction and Sharing

- It is an extended machine providing abstraction of the hardware
  - Hides the messy details which must be performed
  - Presents user with a virtual machine, easier to use
- It is a resource manager
  - Time on CPU is shared among multiple users/programs
  - Space in memory and on disks is shared among multiple users/programs

Pentium Architecture



Abstractions in OS

Hardware

- Disks
- Memory
- Processors
- Network
- Monitor
- Keyboard
- Mouse

OS abstraction

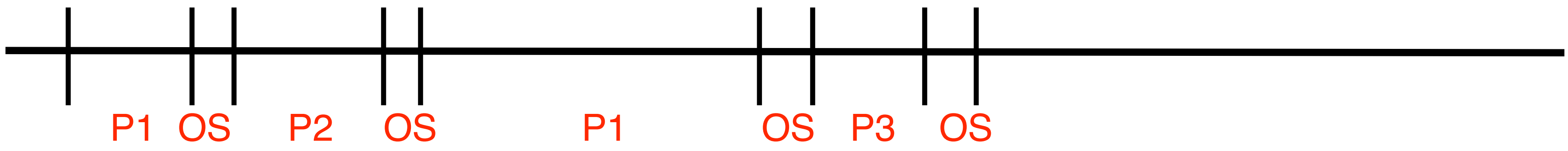
- Files
- Programs
- Threads / Processes
- Communication
- Windows and GUI
- Input
- Locator

Sharing of Memory

Issues

- Allocation schemes
- Protection from each other
- Protecting OS code
- Translating logical addresses to physical
- Swapping programs
- What if physical memory is small: Virtual memory

Timesharing



- At any point, only one program can run on CPU
- Context switch: changing the program that has CPU
- When to switch (goal: to optimize the CPU usage)
  - When a program terminates
  - When a program has run “long enough”
  - When a program executes a system call or waits for I/O
  - When an external interrupt arrives (e.g. mouse click)
- OS must do all the book-keeping necessary for context switch, with minimum number of instructions

Challenges in OS

Why can't Microsoft still get rid of all bugs in Windows ?

- Performance is critical
  - How to reduce the memory and time overhead due to OS
- Synchronization and deadlocks due to shared resources
- Scheduling of multiple programs
  - Fairness, response time, real-time applications
- Memory management
  - Virtual memory, paging, segmentation
- Security and Protection
  - Authorization, authentication, viruses
- Interrupt management and error handling
- Marketability and backward compatibility

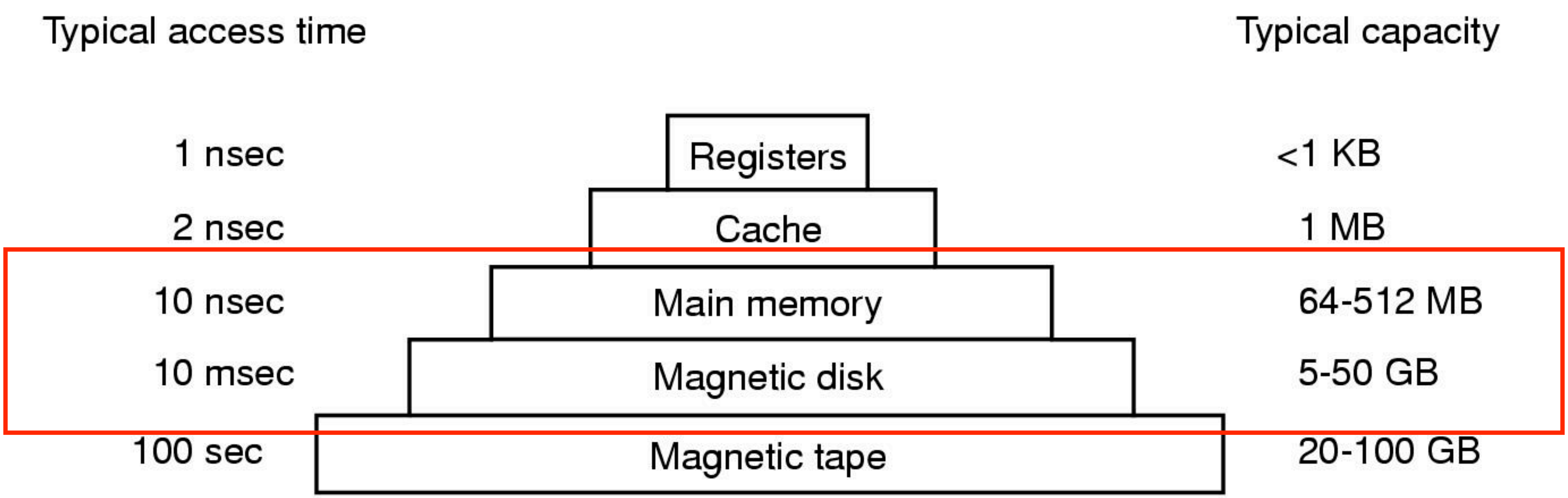
How does OS work?

- OS gets control of the CPU repeatedly
- Let's look at two typical scenarios to get a glimpse of how things work (we will get a more accurate and detailed understanding as the course progresses)
- Basic knowledge about computer architecture is essential ! (Read Sec 1.4 to review CSE 240)

Inside a CPU

- State of a running program
  - Registers
  - Program counter (PC)
  - Stack pointer
  - Program status word (PSW)
- Key distinction in PSW: user mode vs kernel (OS) mode
- Key instruction for OS calls: TRAP (switch to kernel mode)
- Many operations (such as accessing I/O devices) are possible only in the kernel mode

Different types of Memory



- ❑ Use of disks unavoidable (permanence and size)
- ❑ Access time is significantly slower for disks

Sample Scenario 1

- Consider a statement to read from a file in a user program P
- User program stores parameters such as file-id, memory-address, number-of-bytes, and system-call number of read, and executes TRAP instruction to invoke OS
- Hardware saves the state of current program, sets the mode-bit in PSW register in CPU to 1, and transfers control to a fixed location in OS code
- OS maintains an internal file table that stores relevant information about all open files

Sample Scenario 1 (continued)

- OS read routine examines the parameters, checks for errors (e.g. file must be open), consults its file table, and determines the disk address from where data is to be retrieved
- then it sets up registers to initiate transfer by the disk controller
- While disk controller is transferring data from disk to memory, OS can suspend current program, and switch to a different program
- When OS routine finishes the job, it stores the status code, and returns control to the user program P (hardware resets mode-bit)
- Note: Disk controller is accessed only by OS code (this is ensured by hardware protection)

Sample Scenario 2

- Consider an assignment x:=y in a program P
- Compiler assigns logical addresses, say Add1 and Add2, for program variables in P's data space
- When P is loaded in memory, OS assigns a physical base address to store P and its data
- Compiled code looks like

  Load (R, Add1); Store (R, Add2)

- While executing Load instruction the hardware translates the logical address Add1 to a physical memory location (this is done by Memory Management Unit MMU)

Sample Scenario 2 (continued)

- However, OS may not keep all of P in memory all the time
- OS maintains an internal table, called page table, that keeps track of which blocks of P are in memory
- If Add1 is not in memory, MMU generates a page fault, and transfers control to OS
- OS examines the cause, and initiates a disk transfer to load in the relevant block of P
- OS needs to decide memory allocation for the block to be fetched (page replacement algorithms)
- While this block is being fetched, P may be suspended using a context switch

Brief History of Operating Systems

- 1940's -- First Computers
- 1950's -- Batch Processing
- 1960's -- Multiprogramming (timesharing)
- 1970's -- Minicomputers & Microprocessors
- 1980's -- Networking, Distributed Systems, Parallel (multiprocessor) Systems
- 1990's and Beyond -- PCs, WWW, Mobile Systems, embedded systems

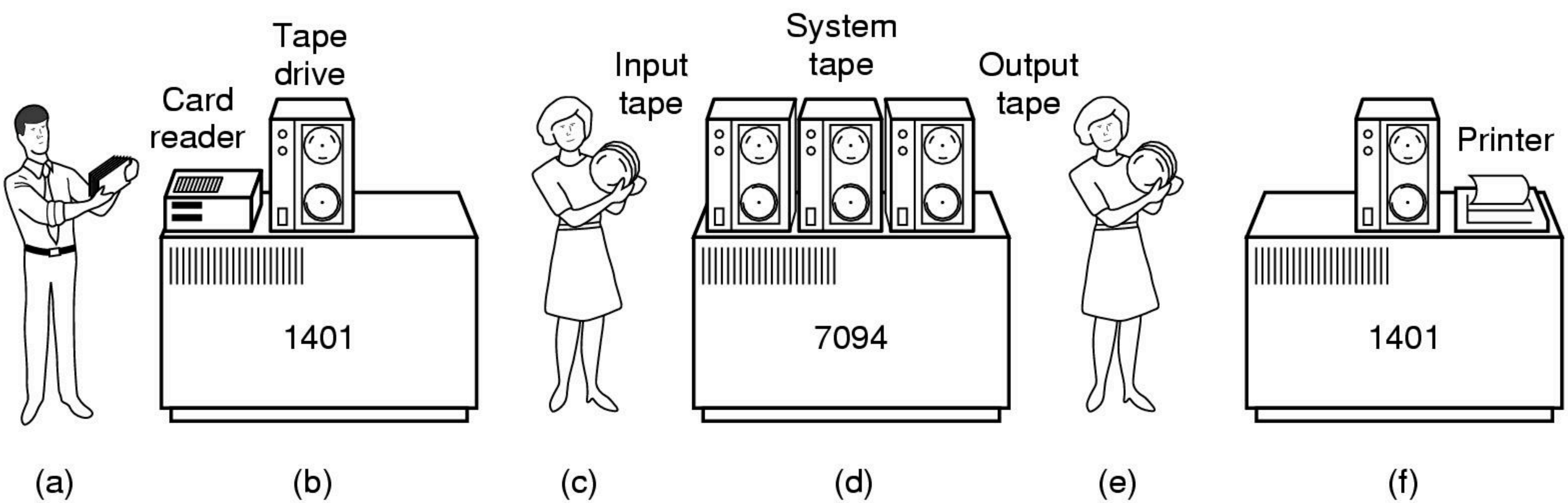
1940's -- First Computers

- Computer dedicated to one user/programmer at a time. Program loaded manually by programmer, using console switches. Debugging using console lights.
- Advantages:
  - Interactive (user gets immediate response)
- Disadvantages:
  - Expensive machine idle most of time, because people are slow.
  - Programming & debugging are tedious.
  - Each program must include code to operate peripherals -- error prone, device dependencies.
- Libraries of subroutines to drive peripherals are example of typical OS service.

1950's -- Batch Processing

- User/programmer submits a deck of cards that describes a job to be executed.
- Jobs submitted by various users are sequenced automatically by a resident monitor.
- Tape drives available for batching of input and spooling of output.
- Advantages:
  - Computer system is kept busier.
- Disadvantages:
  - No longer interactive; longer turnaround time.
  - CPU is still idle for I/O-bound jobs.
- OS issues -- command processor (JCL), protection of resident monitor from user programs, loading of user programs after monitor.

Typical Batch System



Early batch system

- bring cards to 1401
- read cards to tape
- put tape on 7094 which does computing
- put tape on 1401 which prints output

1960's -- **Multiprogramming** (timesharing)

- The advent of the I/O processor made simultaneous I/O and CPU processing possible.
- CPU is multiplexed (shared) among a number of jobs -- while one job waiting for I/O, another can use CPU.
- Advantages:
  - Interactiveness is restored.
  - CPU is kept busy.
- Disadvantages:
  - Hardware and O.S. required become significantly more complex.
- **Timesharing** - switch CPU among jobs for pre-defined time interval
- Most O.S. issues arise from trying to support multiprogramming -- CPU scheduling, deadlock, protection, memory management, virtual memory, etc.
- CTSS (Compatible Time Sharing System), Multics

1970's - Minicomputers & Microprocessors

- Trend towards many small to mid-range personal computers, rather than a single mainframe.
- Early minicomputers and microprocessors were small, so there was some regression to earlier OS ideas.
  - e.g. DOS on PC is still essentially a batch system similar to those used in 1960, with some modern OS ideas thrown in (e.g., hierarchical file system).
- This trend changing rapidly because of powerful new microprocessors.
- Also, the user interface (GUI) became more important.
- **UNIX**, DOS

1980's - Networking

- Powerful workstations (e.g., PDP, VAX, Sunstations, etc.)
- Local area networks (e.g., Ethernet, Token ring) and long-distance network (Arpanet)
- Networks organized with clients and servers
- Decentralization of computing requires more communication (e.g., resource sharing)
- O.S. issues -- network communication protocols, data encryption, security, reliability, consistency of distributed data
- Real-Time Systems – timing constraints, deadlines, QoS (quality of service)

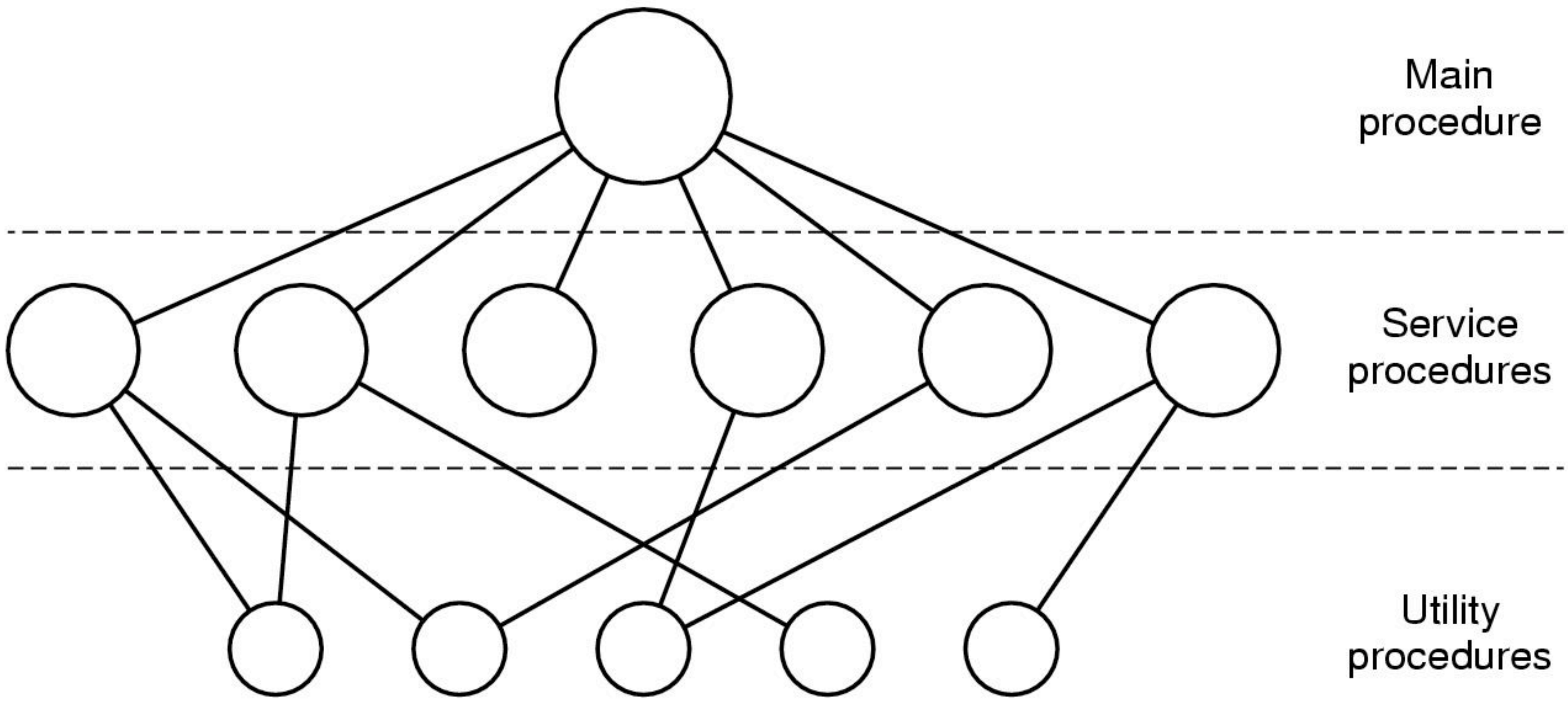
1990's and Beyond

- Parallel Computing (tera-flops)
- Powerful PCs, Multimedia computers
- High-speed, long-distance communication links to send large amounts of data, including graphical, audio and video
- World Wide Web
- Electronic notebooks and PDAs using wireless communication technologies
- Embedded computers: medical devices, cars, smartcards
- O.S. issues -- Large heterogeneous systems, mobile computing, utilization of power, security, etc.

Operating System Structure

- Monolithic Systems
- Layered Systems
- Virtual Machines
- Client-Server Model

Operating System Structure (1)



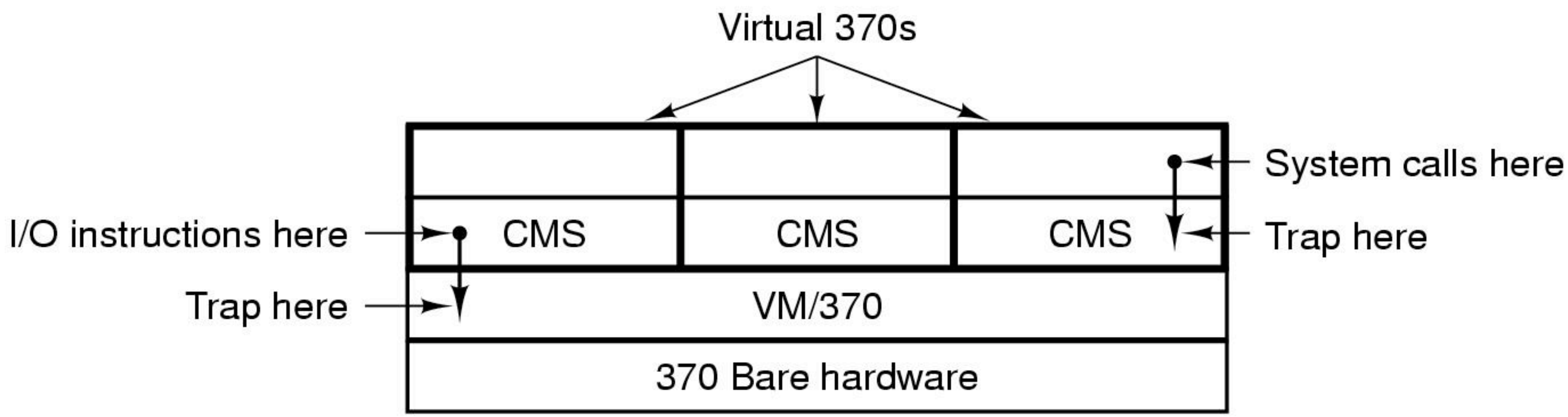
Simple structuring model for a monolithic system

Operating System Structure (2)

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

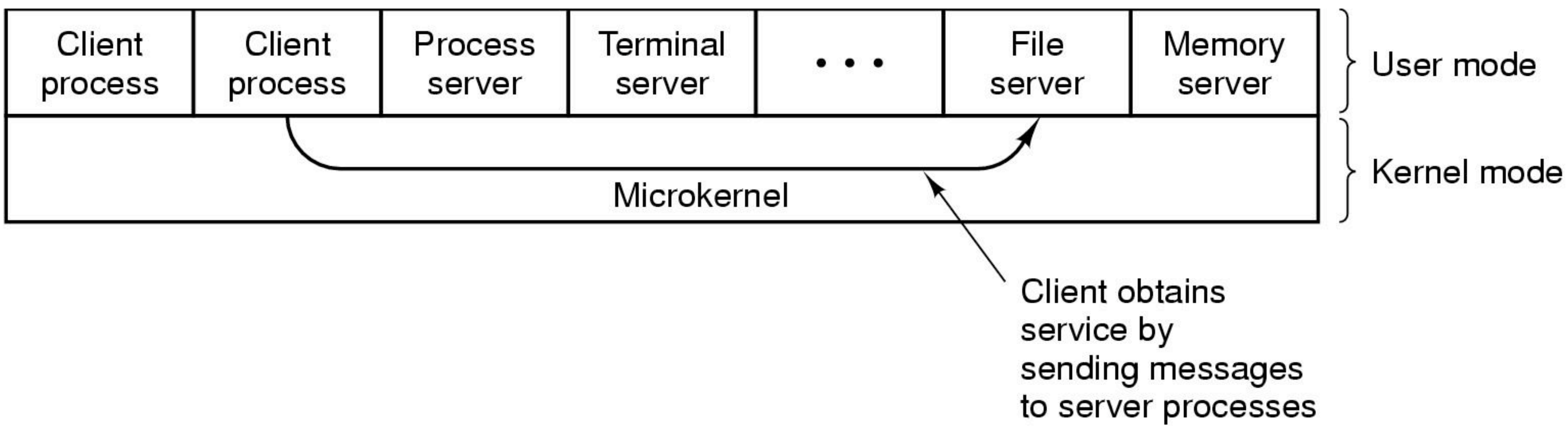
Structure of the THE operating system

Operating System Structure (3)



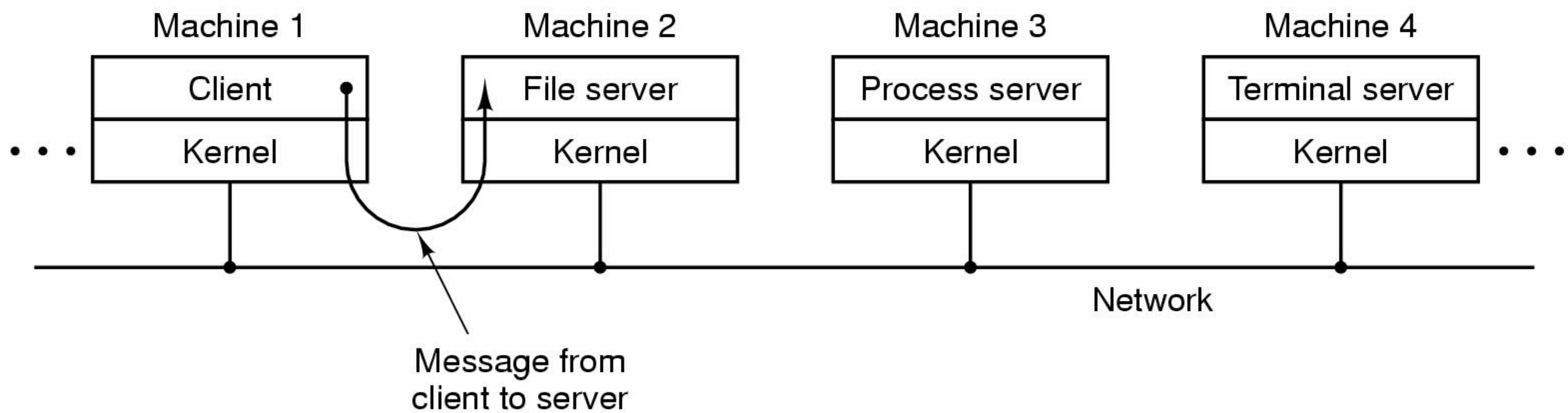
Structure of VM/370 with CMS

Operating System Structure (4)



The client-server model

Operating System Structure (5)



The client-server model in a distributed system