

# Chapter 1 ~ 3 : Operating System Overview

王振傑 (Chen-Chieh Wang)  
ccwang@mail.ee.ncku.edu.tw

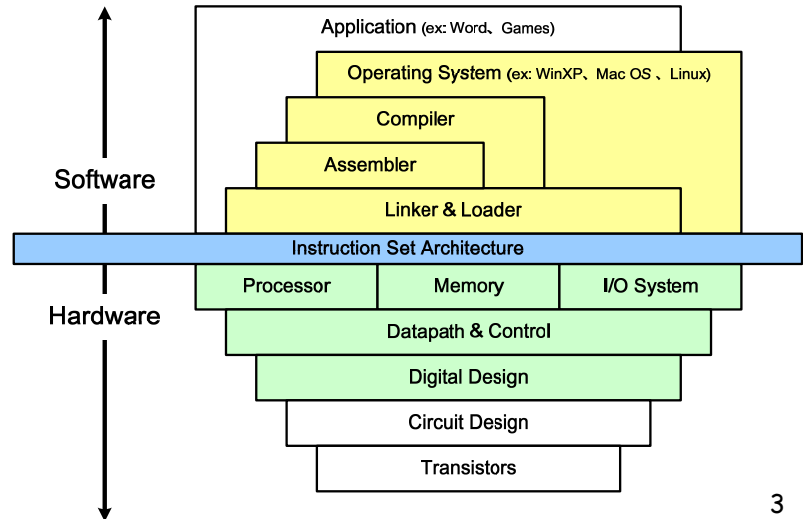
System Programming, Spring 2010

## Outline

- ✦ Introduction
  - What Is an Operating System?
  - What Operating System Do?
- ✦ Computer-System Structures
  - Computer-System Operation
  - I/O Structure
  - Hardware Protection
- ✦ Operating-System Structures
  - System Components
  - System Structure
  - Virtual Machines

# Computer System

- ⊕ User
- ⊕ Application program
- ⊕ Operating system
- ⊕ Hardware



3

# 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.

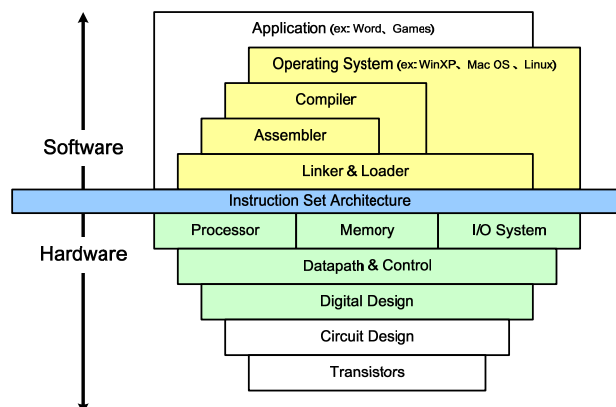
4

# 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.)

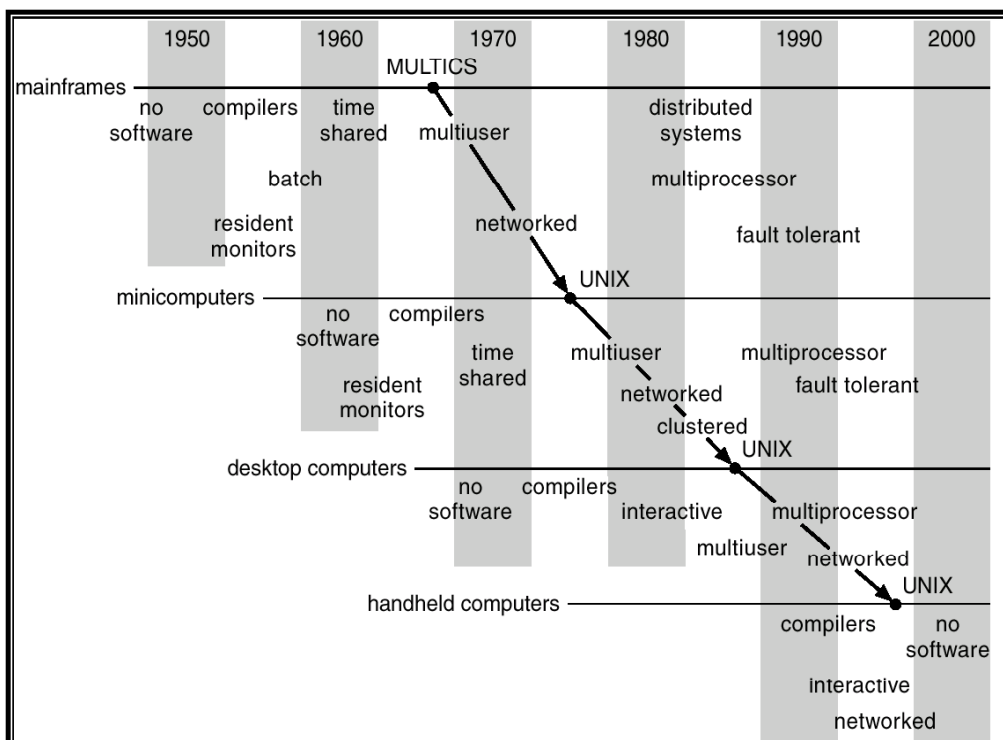
- ✦ Top-down view
  - The OS as an **Extended Machine (Virtual Machine)**
  
- ✦ Bottom-up view
  - The OS as a **Resource Manager**



## 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

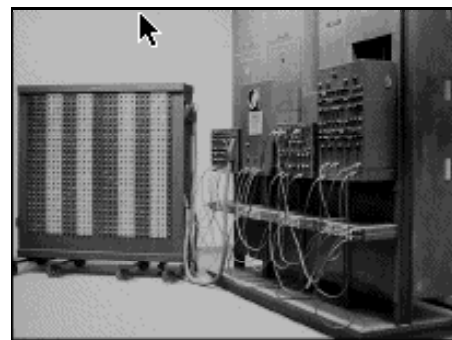
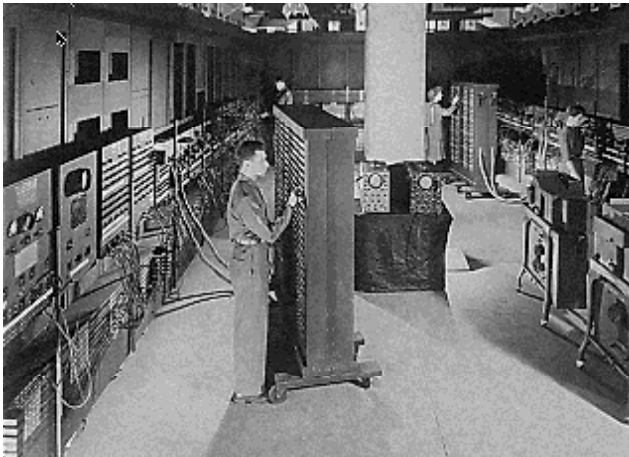
## Migration of Operating-System Concepts and Features



# History of Operating Systems

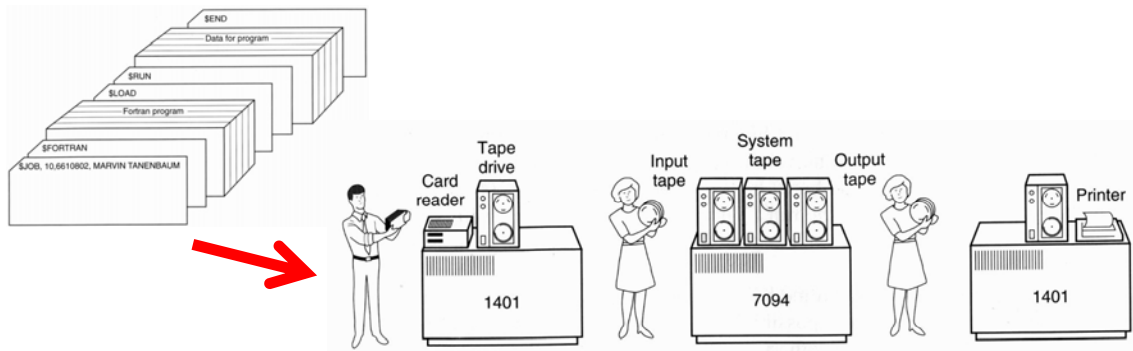
- ✦ The 1<sup>st</sup> generation (1945-55) : Vacuum Tubes and Plugboards
- ✦ The 2<sup>nd</sup> generation (1955-65) : Transistors and Batch Systems
  - Batch System
- ✦ The 3<sup>rd</sup> generation (1965-1980) : ICs and Multiprogramming
  - Multiprogramming
  - Spooling (Simultaneous Peripheral Operation On Line)
  - Timesharing
  - MULTICS (MULTIplexed Information and Computing Service)
  - UNIX, MINIX, Linux → IEEE POSIX (Portable OS Interface for Unix) Standard
- ✦ The 4<sup>th</sup> generation (1980-present) : Personal Computers
  - CP/M (Control Program for Microcomputers)
  - DOS (Disk Operating System)
  - MS-DOS (MicroSoft Disk Operating System)
  - GUI (Graphical User Interface)

## Dawn of time ENIAC: (1945–1955)



- ✦ “The machine designed by Drs. Eckert and Mauchly was a monstrosity. When it was finished, the ENIAC filled an entire room, weighed thirty tons, and consumed two hundred kilowatts of power.”
- ✦ <http://ei.cs.vt.edu/~history/ENIAC.Richey.HTML>

# Batch System



IBM 7094

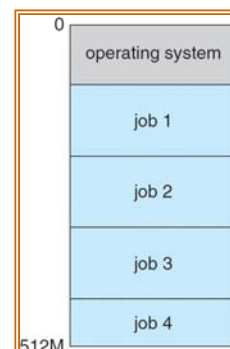


IBM 1401

# Multiprogramming

## ⊕ Multiprogramming 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

- ✦ **Time-sharing (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

13

System Programming, Spring 2010

## A Multics System (Circa 1976)

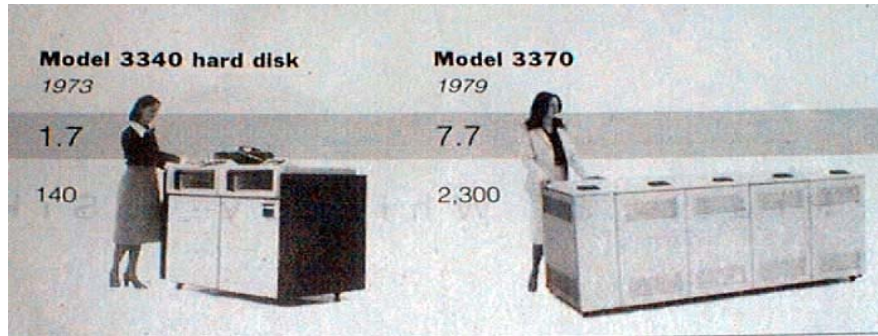


- ✦ The 6180 at MIT IPC, skin doors open, circa 1976:
  - “We usually ran the machine with doors open so the operators could see the AQ register display, which gave you an idea of the machine load, and for convenient access to the EXECUTE button, which the operator would push to enter BOS if the machine crashed.”
- ✦ <http://www.multicians.org/multics-stories.html>

14

System Programming, Spring 2010

## Early Disk History



**1973:**  
1.7 Mbit/sq. in  
140 MBytes

**1979:**  
7.7 Mbit/sq.  
in 2,300 MBytes

**Contrast: Seagate 1TB,  
164 GB/SQ in, 3½ in disk,  
4 platters**



15

System Programming, Spring 2010

## Outline

- ✦ Introduction
  - What Is an Operating System?
  - What Operating System Do?
- ✦ Computer-System Structures
  - Computer-System Operation
  - I/O Structure
  - Hardware Protection
- ✦ Operating-System Structures
  - System Components
  - System Structure
  - Virtual Machines

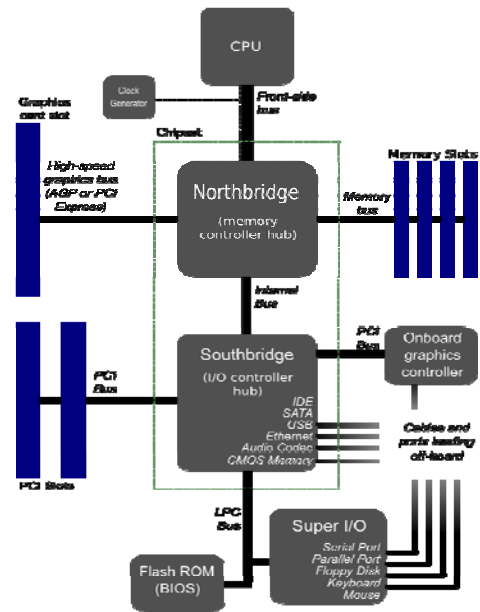
16

System Programming, Spring 2010



# Computer Startup

- ✦ bootstrap program is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

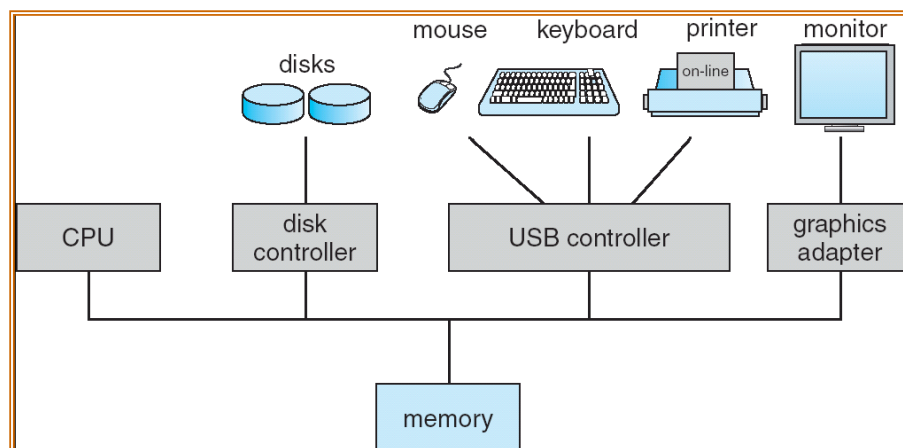


Source : [http://en.wikipedia.org/wiki/Front\\_side\\_bus](http://en.wikipedia.org/wiki/Front_side_bus) 17

System Programming, Spring 2010

# 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

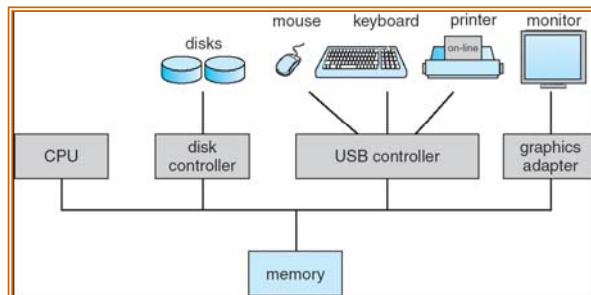


18

System Programming, Spring 2010

## 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**.



19

System Programming, Spring 2010

## Common Functions of Interrupts

- ✦ Interrupt transfers control to the **interrupt service routine (ISR)** 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**.

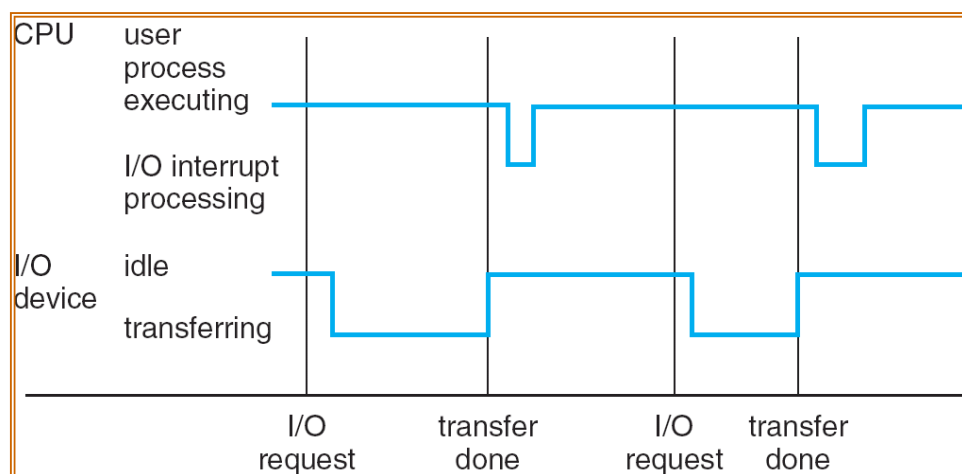
20

System Programming, Spring 2010

# 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

# Interrupt Timeline



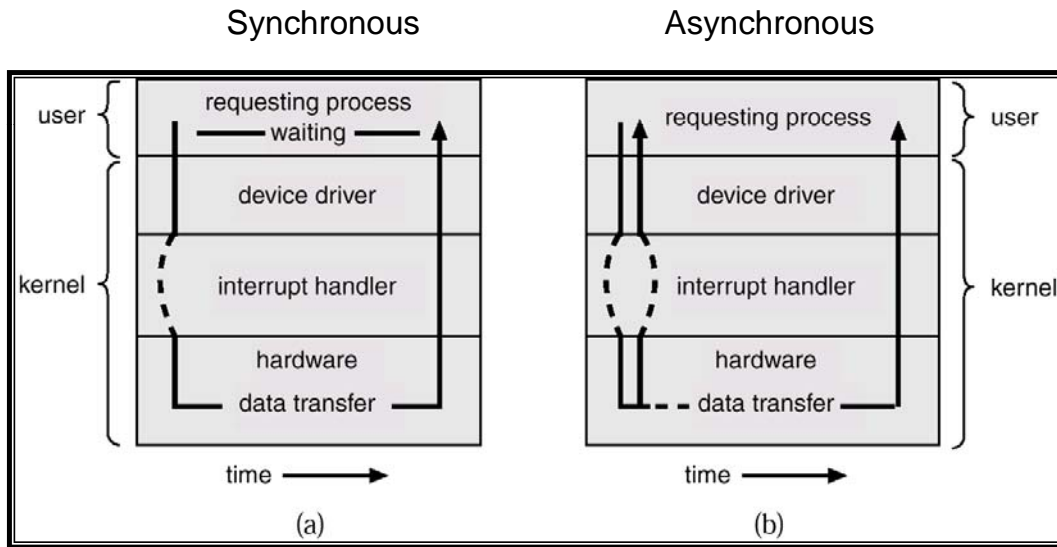
# Outline

- ✦ Introduction
  - What Is an Operating System?
  - What Operating System Do?
- ✦ Computer-System Structures
  - Computer-System Operation
  - I/O Structure
  - Hardware Protection
- ✦ Operating-System Structures
  - System Components
  - System Structure
  - Virtual Machines

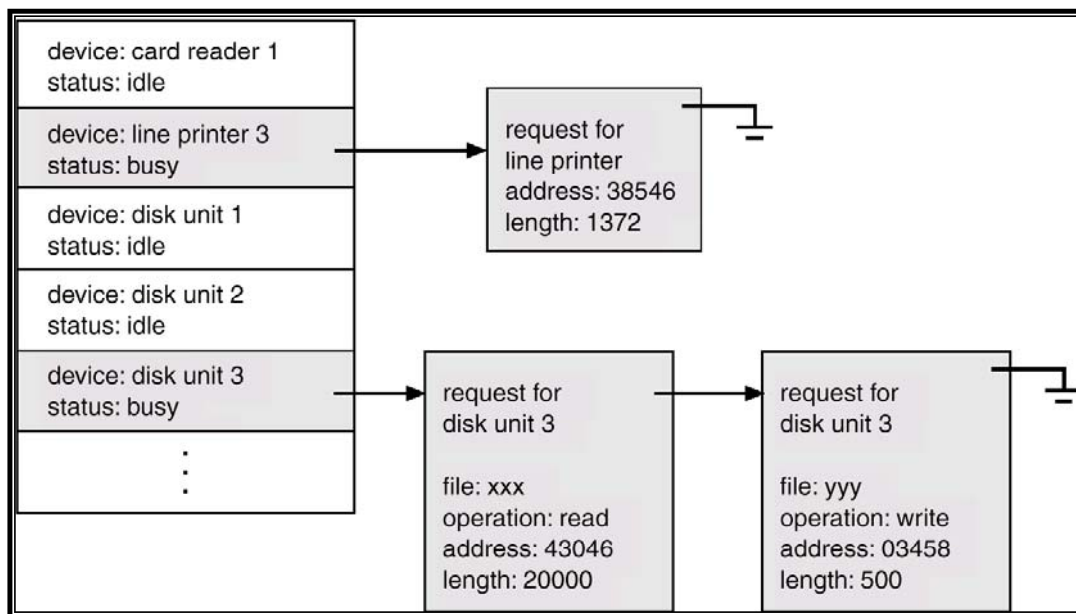
# I/O Structure

- ✦ After I/O starts, control returns to user program only **upon** I/O completion.
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access).
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing.
- ✦ After I/O starts, control returns to user program **without** waiting for I/O completion.
  - *System call*– request to the operating system to allow user to wait for I/O completion.
  - *Device-status table* contains entry for each I/O device indicating its type, address, and state.
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

## Two I/O Methods



## Device-Status Table



# 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.

# Outline

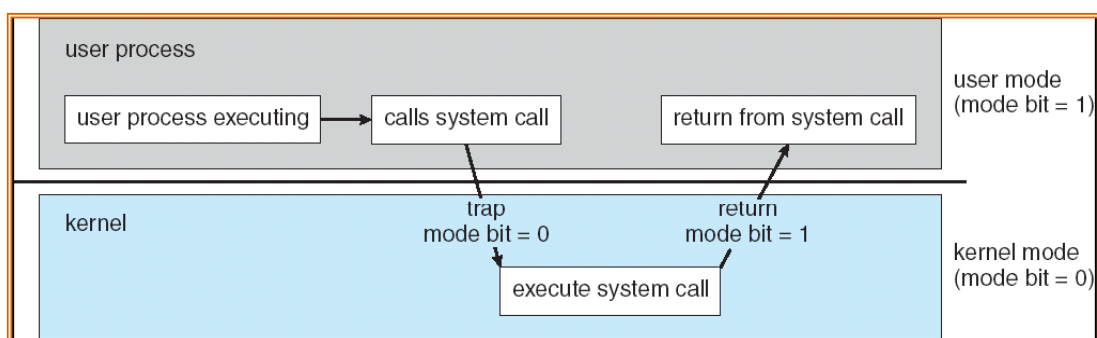
- ✦ Introduction
  - What Is an Operating System?
  - What Operating System Do?
- ✦ Computer-System Structures
  - Computer-System Operation
  - I/O Structure
  - Hardware Protection
- ✦ Operating-System Structures
  - System Components
  - System Structure
  - Virtual Machines

# Operating-System Operations

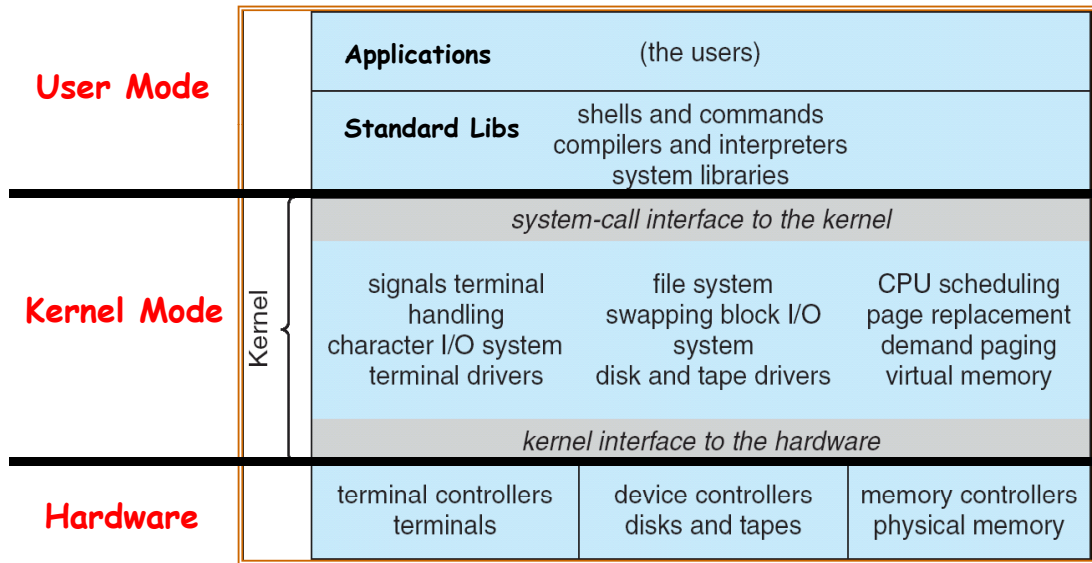
- ✦ **Interrupt** driven by hardware
- ✦ Software error or request creates **exception** or **trap**
  - Division by zero, request for operating system service
- ✦ Other process problems include infinite loop, processes modifying each other or the operating system
- ✦ **Dual-mode** operation allows OS 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

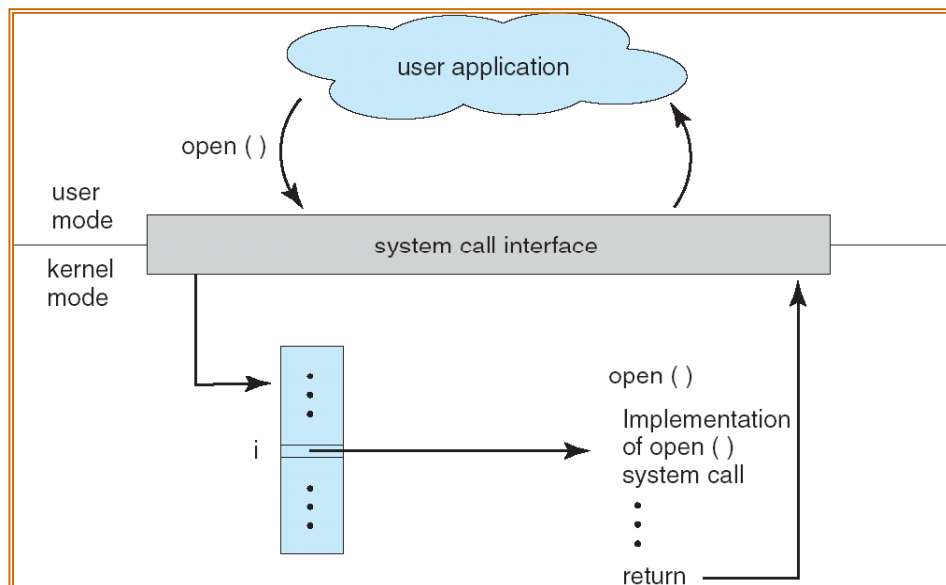
- ✦ **Timer** to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



# UNIX System Structure



# System Calls (What is the API)





# Outline

- ⊕ Introduction
  - What Is an Operating System?
  - What Operating System Do?
- ⊕ Computer-System Structures
  - Computer-System Operation
  - I/O Structure
  - Hardware Protection
- ⊕ Operating-System Structures
  - System Components
  - System Structure
  - Virtual Machines

# System Components

- ⊕ Process Management (*Chapter 4~8*)
- ⊕ Memory Management (*Chapter 9~10*)
- ⊕ File Management
- ⊕ I/O-System Management
- ⊕ Storage Management
- ⊕ Networking
- ⊕ Protection System
- ⊕ Command-Interpreter System

## Process Management

- ✦ 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

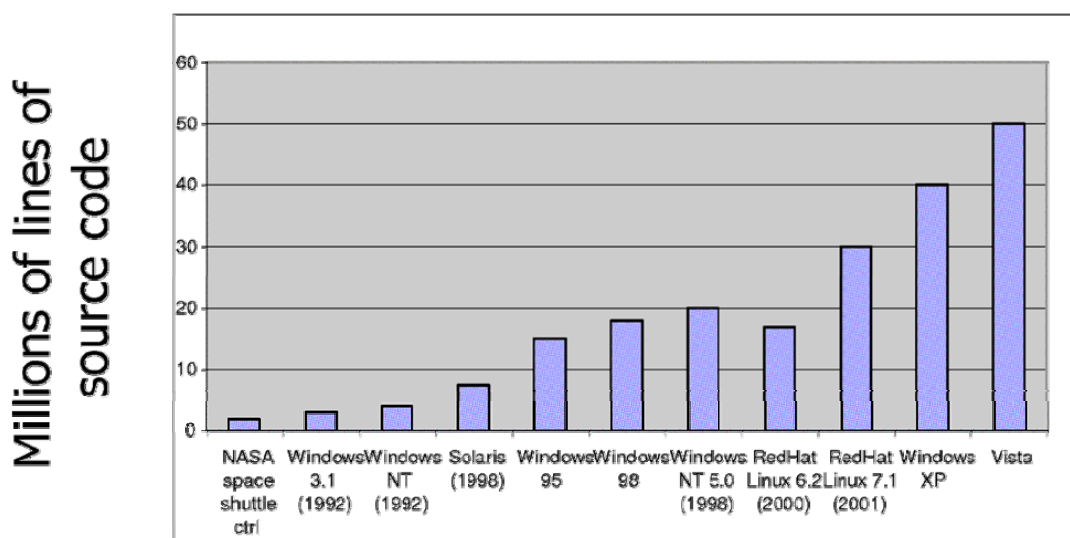
## Memory Management

- ✦ All **data** in memory before and after processing
- ✦ All **instructions** in memory in order to execute
- ✦ Memory management determines what is in memory 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

# Outline

- ✦ Introduction
  - What Is an Operating System?
  - What Operating System Do?
- ✦ Computer-System Structures
  - Computer-System Operation
  - I/O Structure
  - Hardware Protection
- ✦ Operating-System Structures
  - System Components
  - **System Structure**
  - Virtual Machines

# Increasing Software Complexity



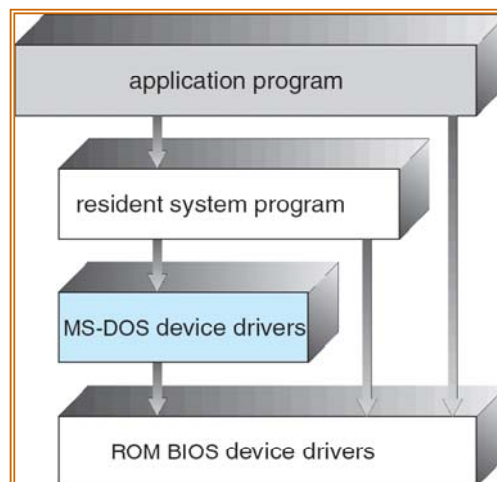
From MIT's 6.033 course

## System Structure

- ✦ A systems as large and complex as a modern operating system must be engineered carefully if it is to function properly and be modified easily
  - **Simple** : Only one or two levels of code
  - **Layered** : Lower levels independent of upper levels
  - **Microkernel** : OS built from many user-level processes
  - **Modular** : Core kernel with Dynamically loadable modules

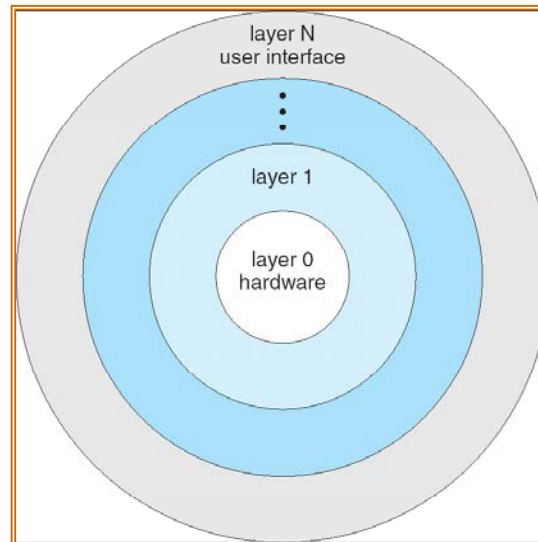
## Simple Structure

- ✦ MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated



## Layered Approach

- ✦ The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the **hardware**; the highest (layer N) is the **user interface**.



41

System Programming, Spring 2010

## Microkernel System Structure

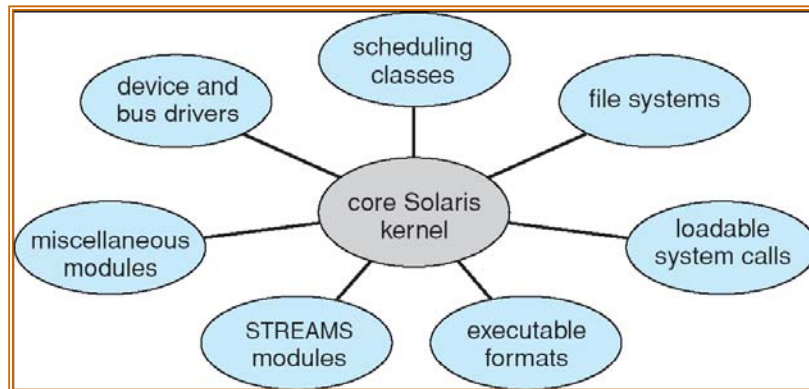
- ✦ Moves as much from the kernel into “*user*” space
- ✦ Communication takes place between user modules using message passing
- ✦ Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- ✦ Detriments:
  - Performance overhead of user space to kernel space communication

42

System Programming, Spring 2010

## Modules-based Structure

- ✦ Most modern operating systems implement modules
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- ✦ Overall, similar to layers but with more flexible



43

System Programming, Spring 2010

## Outline

- ✦ Introduction
  - What Is an Operating System?
  - What Operating System Do?
- ✦ Computer-System Structures
  - Computer-System Operation
  - I/O Structure
  - Hardware Protection
- ✦ Operating-System Structures
  - System Components
  - System Structure
  - **Virtual Machines**

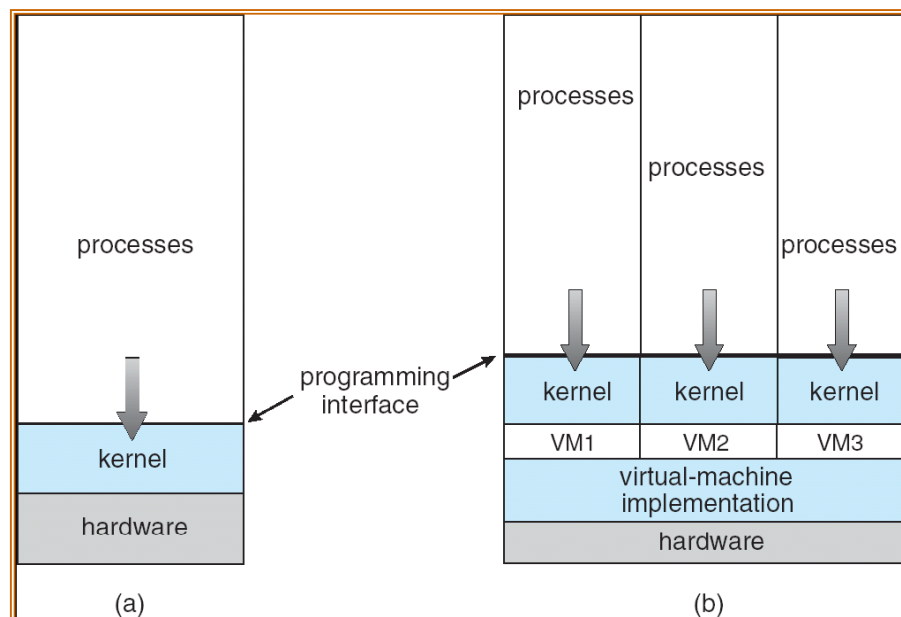
44

System Programming, Spring 2010

# Virtual Machines

- ✦ A *virtual machine* takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware
- ✦ A virtual machine provides an interface *identical* to the underlying bare hardware
- ✦ The operating system creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory

# Virtual Machines (Cont.)



## Virtual Machines (Cont.)

- ✦ The resources of the physical computer are shared to create the virtual machines
  - CPU scheduling can create the appearance that users have their own processor
  - Spooling and a file system can provide virtual card readers and virtual line printers
  - A normal user time-sharing terminal serves as the virtual machine operator's console

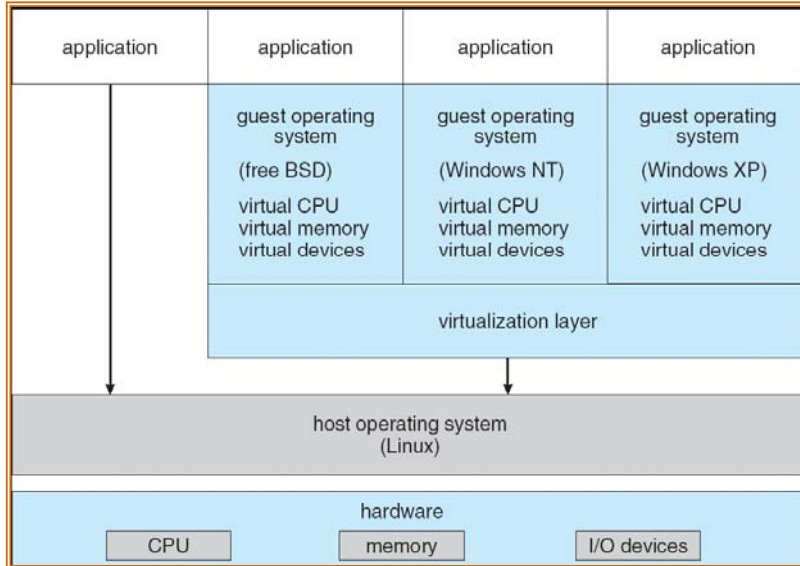
## Virtual Machines (Cont.)

- ✦ The virtual-machine concept provides complete **protection** of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- ✦ A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.
- ✦ The virtual machine concept is difficult to implement due to the effort required to provide an *exact* duplicate to the underlying machine

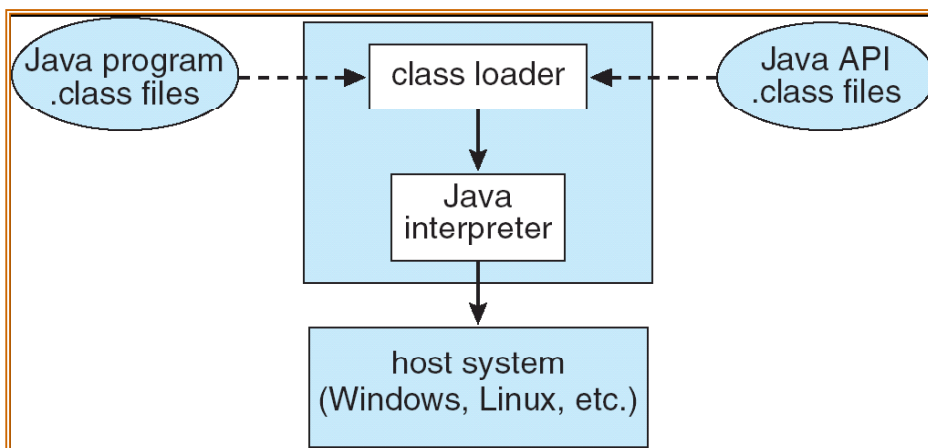


## Virtual Machines (Cont.)

- ✦ Useful for OS development
  - When OS crashes, restricted to one VM
  - Can aid testing programs on other OSs



## The Java Virtual Machine



# Why Study Operating Systems?

- ✦ **Learn how to build complex systems:**
  - How can you manage complexity for future projects?
- ✦ **Engineering issues:**
  - Why is the web so slow sometimes? Can you fix it?
  - What features should be in the next mars Rover?
  - How do large distributed systems work? (Kazaa, etc)
- ✦ **Buying and using a personal computer:**
  - Why different PCs with same CPU behave differently
  - How to choose a processor (Opteron, Itanium, Celeron, Pentium, Hexium)? [ Ok, made last one up ]
  - Should you get Windows XP, 2000, Linux, Mac OS ...?
  - Why does Microsoft have such a bad name?
- ✦ **Business issues:**
  - Should your division buy thin-clients vs PC?
- ✦ **Security, viruses, and worms**
  - What exposure do you have to worry about?

# OS Systems Principles

- ✦ **OS as illusionist:**
  - Make hardware limitations go away
  - Provide illusion of dedicated machine with infinite memory and infinite processors
- ✦ **OS as government:**
  - Protect users from each other
  - Allocate resources efficiently and fairly
- ✦ **OS as complex system:**
  - Constant tension between simplicity and functionality or performance
- ✦ **OS as history teacher**
  - Learn from past
  - Adapt as hardware tradeoffs change