# Operating System Concepts

Tei-Wei Kuo

ktw@csie.ntu.edu.tw

Dept. of Computer Science and Information Engineering

National Taiwan University

# Contents

# Chapter 1. Introduction

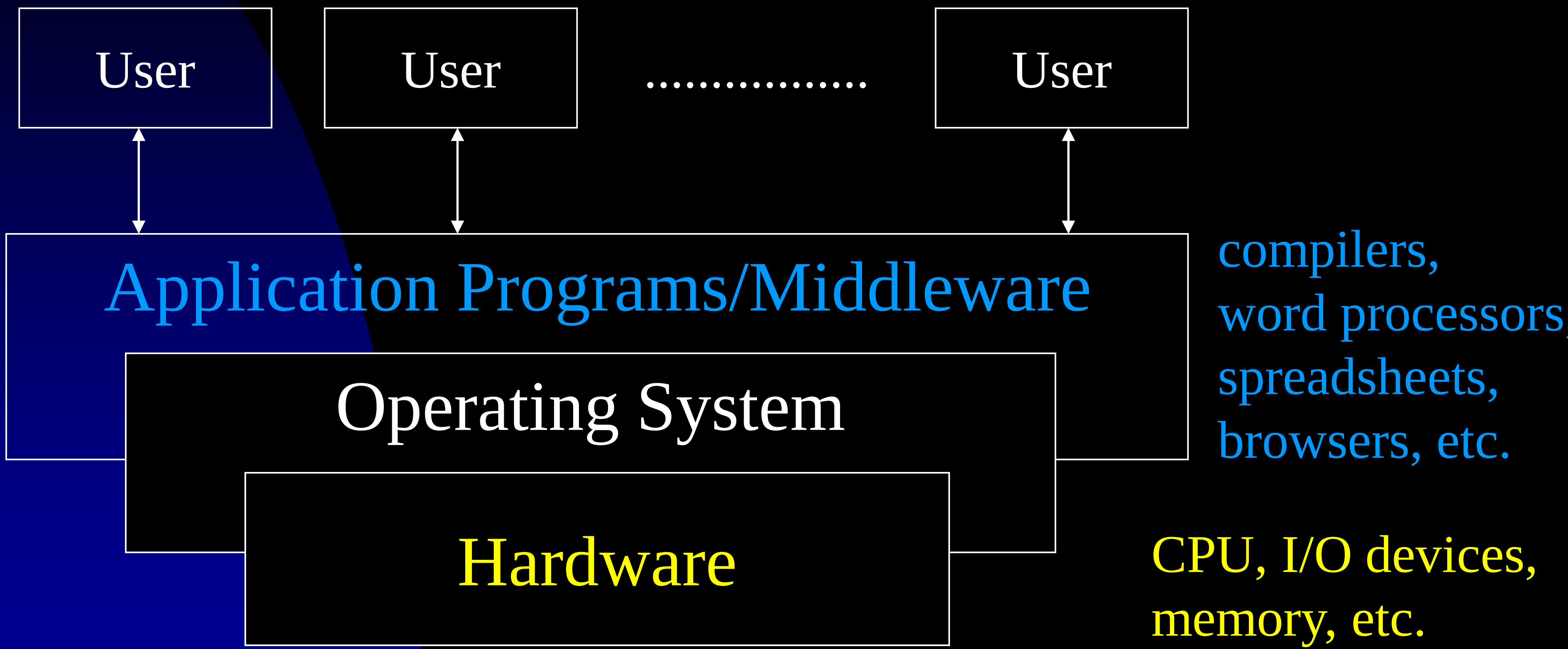
# Introduction

- What is an Operating System?
  - A basis for application programs
  - An intermediary between users and hardware

- Amazing variety
  - Mainframe, personal computer (PC), handheld computer, embedded computer without any user view

Convenient vs Efficient

# Computer System Components




- OS – a government/environment provider

# User View

- The user view of the computer varies by the interface being used!
- Examples:
  - Personal Computer → Ease of use
  - Mainframe or minicomputer → maximization of resource utilization
    - Efficiency and fair share
  - Workstations → compromise between individual usability & resource utilization
  - Handheld computer → individual usability
  - Embedded computer without user view → run without user intervention

# System View

- A Resource Allocator
  - CPU time, Memory Space, File Storage, I/O Devices, Shared Code, Data Structures, and more
- A Control Program
  - Control execution of user programs
  - Prevent errors and misuse
- OS definitions – US Dept.of Justice against Microsoft in 1998
  - The stuff shipped by vendors as an OS
  - Run at all time

# System Goals

- Two Conflicting Goals:
  - Convenient for the user!
  - Efficient operation of the computer system!
- We should
  - recognize the influences of operating systems and computer architecture on each other
  - and learn why and how OS's are by tracing their evolution and predicting what they will become!

# UNIX Architecture

us user user user user user user

User interface

Shells, compilers, X, application programs, etc.

System call interface

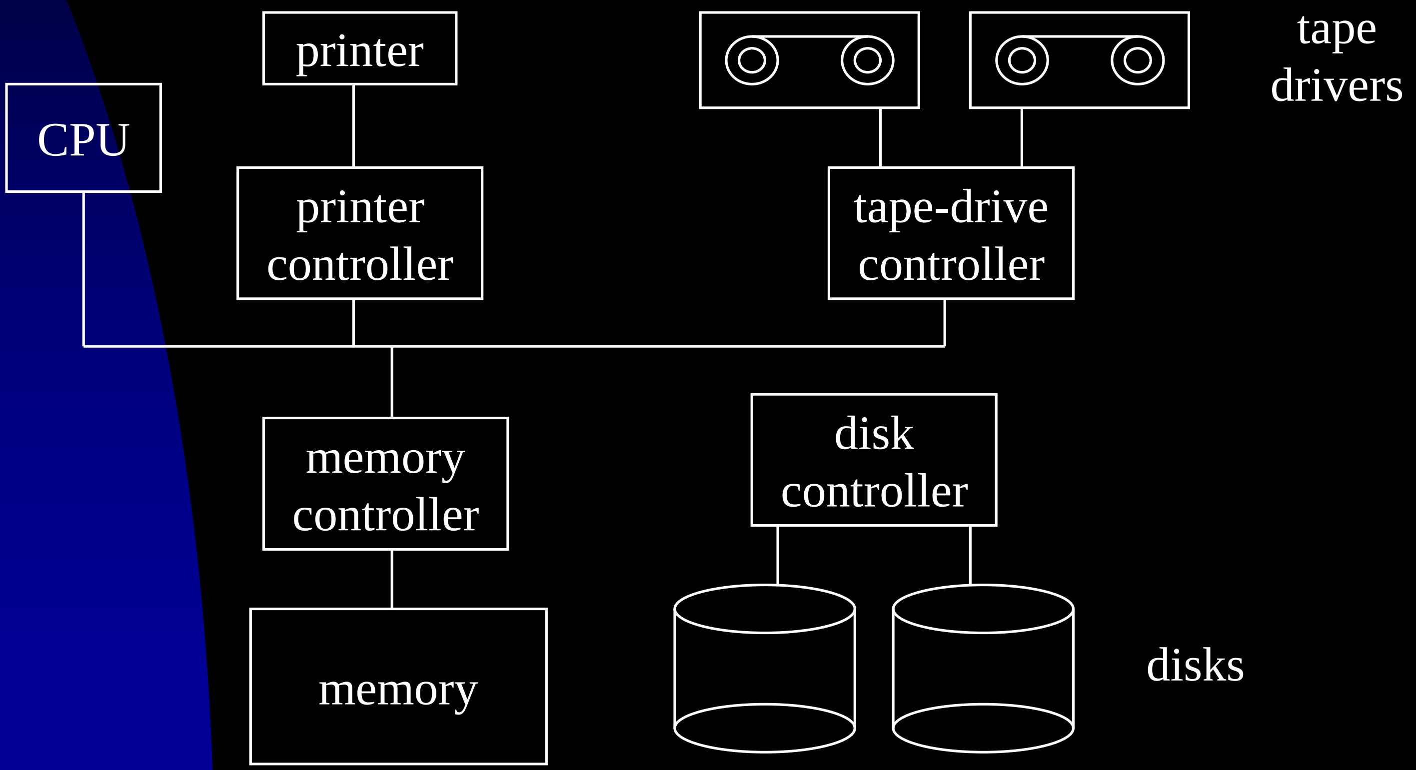
CPU scheduling, signal handling,
virtual memory, paging, swapping,
file system, disk drivers, caching/buffering, etc.

Kernel interface to the hardware

terminal controller, terminals,
physical memory, device controller,
devices such as disks, memory, etc.

# Computer-System Organization

- Objective: General knowledge of the structure of a computer system.



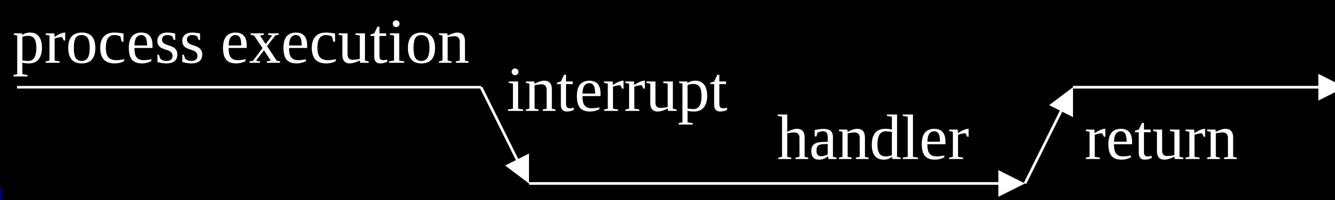

- Device controllers: synchronize and manage access to devices.

# Booting

- Bootstrap program:
  - Initialize all aspects of the system, e.g., CPU registers, device controllers, memory, etc.
  - Load and run the OS
- Operating system: run *init* to initialize system processes, e.g., various daemons, login processes, after the kernel has been bootstrapped. (/etc/rc* & init or /sbin/rc* & init)

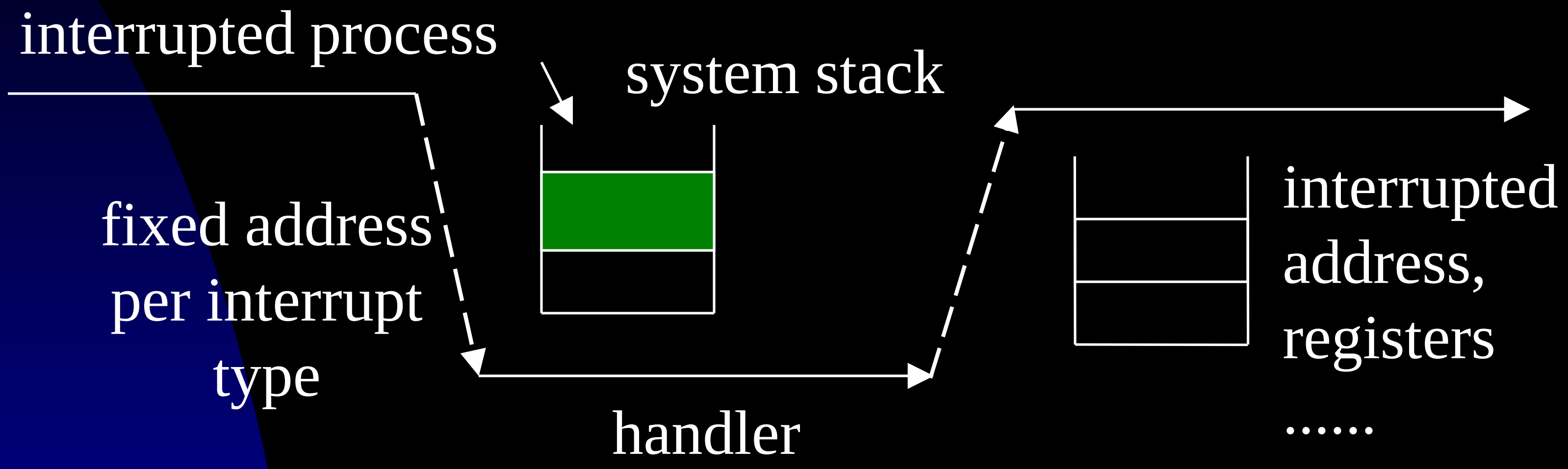
# Interrupt

- Hardware interrupt, e.g. services requests of I/O devices
- Software interrupt, e.g. signals, invalid memory access, division by zero, system calls, etc – (trap)




- Procedures: generic handler or interrupt vector (MS-DOS,UNIX)

# Interrupt Handling Procedure

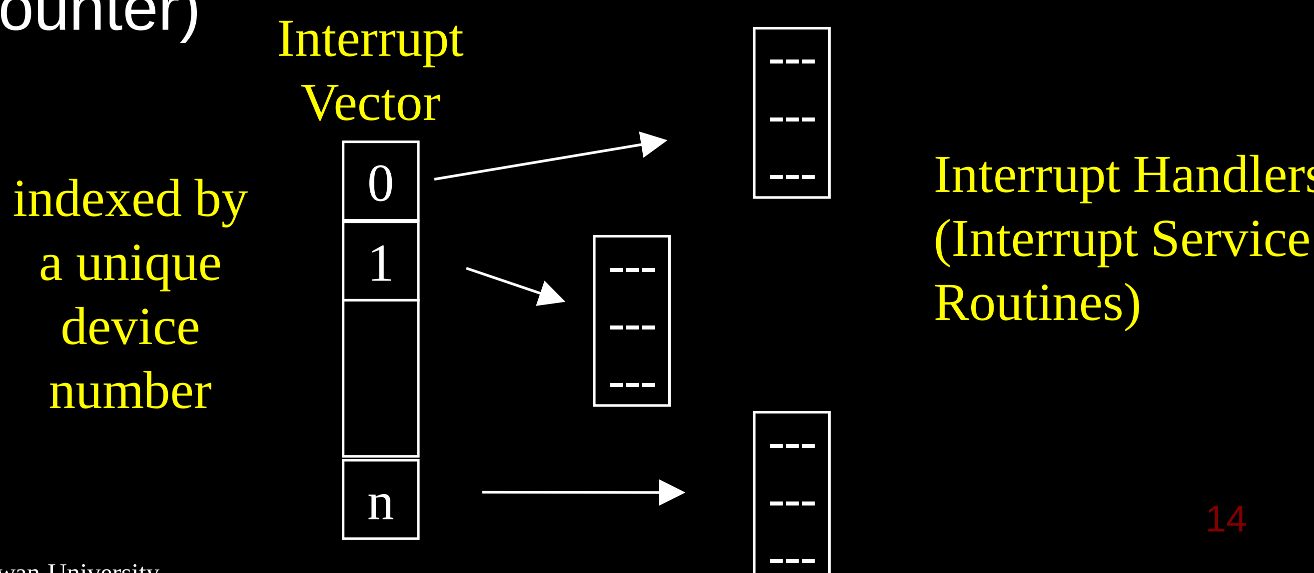



- Saving of the address of the interrupted instruction: fixed locations or stacks
- Interrupt disabling or enabling issues: lost interrupt?!

prioritized interrupts → masking

# Interrupt Handling Procedure

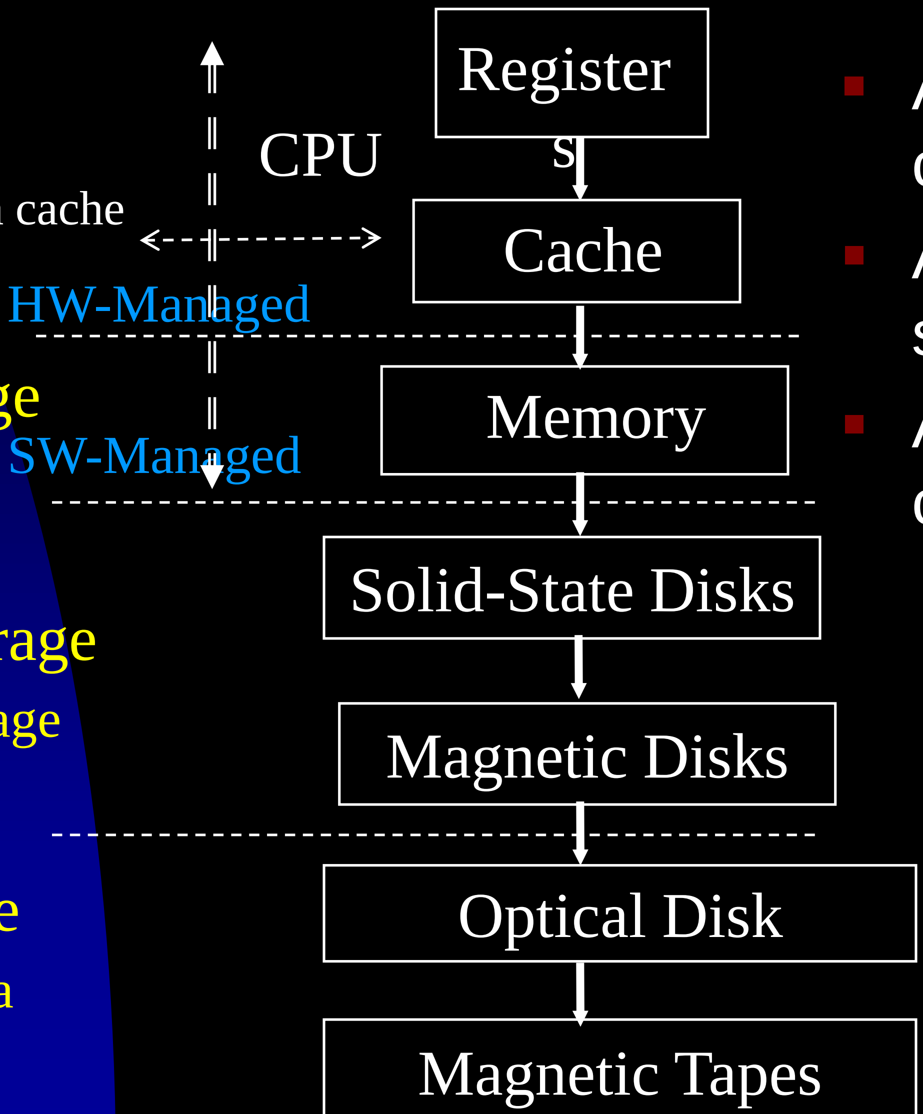
- Interrupt Handling
  - → Save interrupt information
  - → OS determine the interrupt type (by polling)
  - → Call the corresponding handlers
  - → Return to the interrupted job by the restoring important information (e.g., saved return addr. → program counter)



# Storage Structure

High hitting rate

- instruction & data cache
- combined cache




Primary Storage

- volatile storage

Secondary Storage

- nonvolatile storage

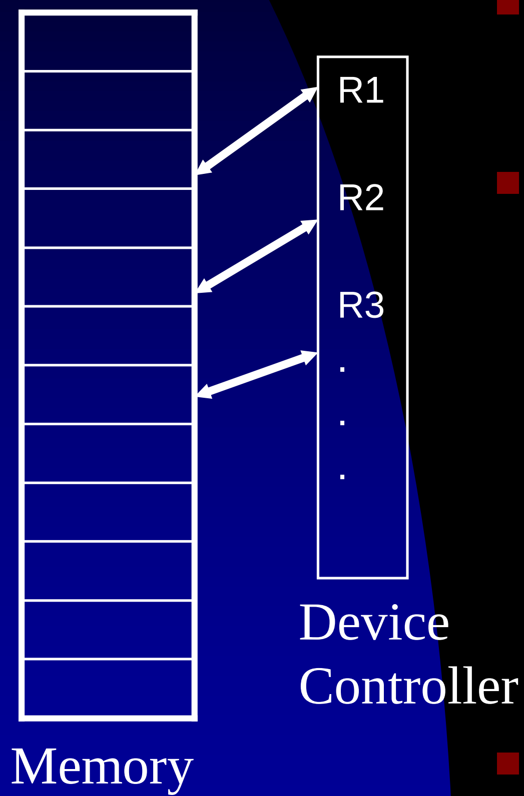
Tertiary Storage

- removable media

- Access time: a cycle
- Access time: several cycles
- Access time: many cycles

* Differences:
Size, Cost,
Speed, Volatility

# Memory

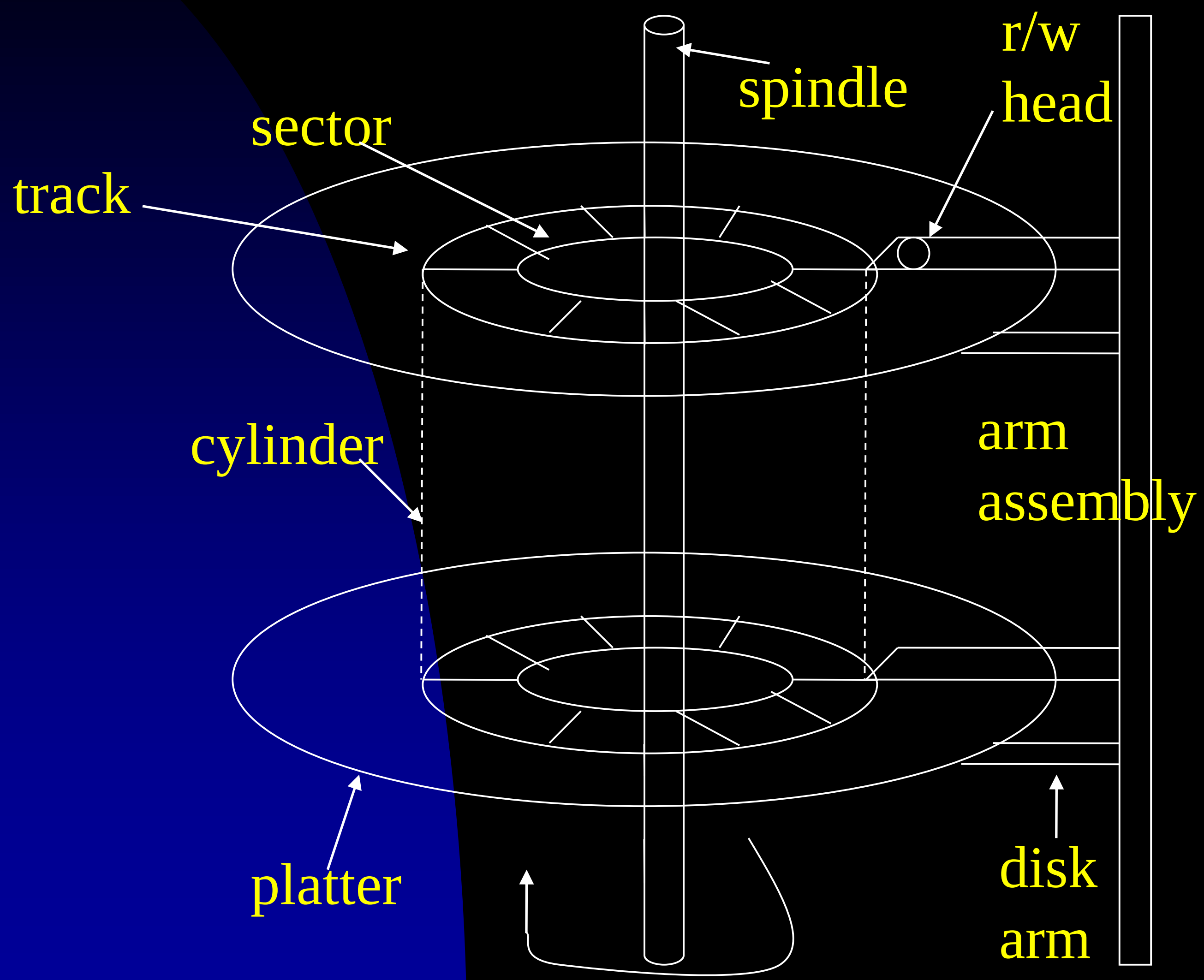



- Processor can have direct access!
- Intermediate storage for data in the registers of device controllers
- Memory-Mapped I/O (PC & Mac)

  (1) Frequently used devices

  (2) Devices must be fast, such as video controller, or special I/O instructions is used to move data between memory & device controller registers
- Programmed I/O – polling
  - or interrupt-driven handling

# Magnetic disks



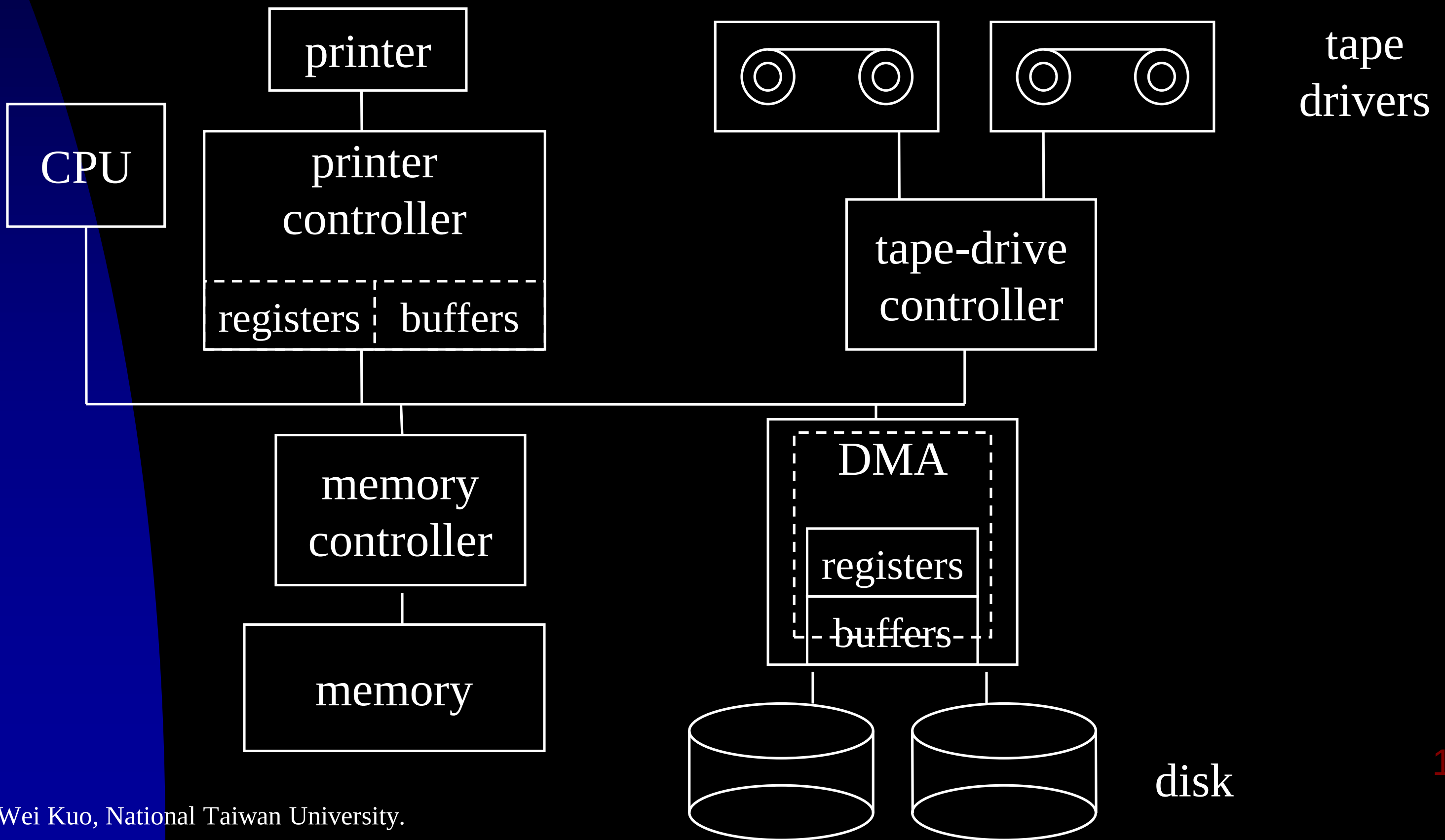

- Transfer Rate
- Random-Access Time
  - Seek time in *x* ms
  - Rotational latency in *y* ms
    - 60~200 times/sec

# Magnetic Disks

- Disks
  - Fixed-head disks:
    - More r/w heads v.s. fast track switching
  - Moving-head disks (hard disk)
  - Primary concerns:
    - Cost, Size, Speed
  - Computer → host controller → disk controller → disk drives (cache ←→ disks)
- Floppy disk
  - slow rotation, low capacity, low density, but less expensive
- Tapes: backup or data transfer bet machines

# I/O Structure

- Device controllers are responsible of moving data between the peripheral devices and their local buffer storages.



# I/O Structure

- I/O operation
  a. CPU sets up specific controller registers within the controller.
  b. Read: devices → controller buffers → memory

     Write: memory → controller buffers → devices
  c. Notify the completion of the operation by triggering an interrupt

# DMA

- Goal: Release CPU from handling excessive interrupts!
  - E.g. 9600-baud terminal

    2-microsecond service / 1000 microseconds

    High-speed device:

    2-microsecond service / 4 microseconds
- Procedure
  - Execute the device driver to set up the registers of the DMA controller.
  - DMA moves blocks of data between the memory and its own buffers.
  - Transfer from its buffers to its devices.
  - Interrupt the CPU when the job is done.

# Single-Processor Systems

- Characteristics: One Main CPU
  - Special-Purpose Processors, e.g., Disk-Controller Microprocessors.
- Examples:
  - Personal Computers (Since 1970's), Mainframes.
- Operating Systems
  - Batching → Multiprogramming → Time-Sharing

# Multiprocessor/Parallel Systems

- Tightly coupled: have more than one processor in close communication sharing computer bus, clock, and sometimes memory and peripheral devices
- Loosely coupled: otherwise
- Advantages
  - Speedup – Throughput
  - Lower cost – Economy of Scale
  - More reliable – Graceful Degradation → Fail Soft (detection, diagnosis, correction)
    - A Tandem or HP-NonStop fault-tolerance solution

# Multiprocessor/Parallel Systems

- Symmetric multiprocessing model: each processor runs an identical copy of the OS
- Asymmetric multiprocessing model: a master-slave relationship
  - ~ Dynamically allocate or pre-allocate tasks
  - ~ Commonly seen in extremely large systems
  - ~ Hardware and software make a difference?
- Trend: the dropping of microporcessor cost

  ➔ OS functions are offloaded to slave processors (back-ends)

# Multiprocessor/Parallel Systems

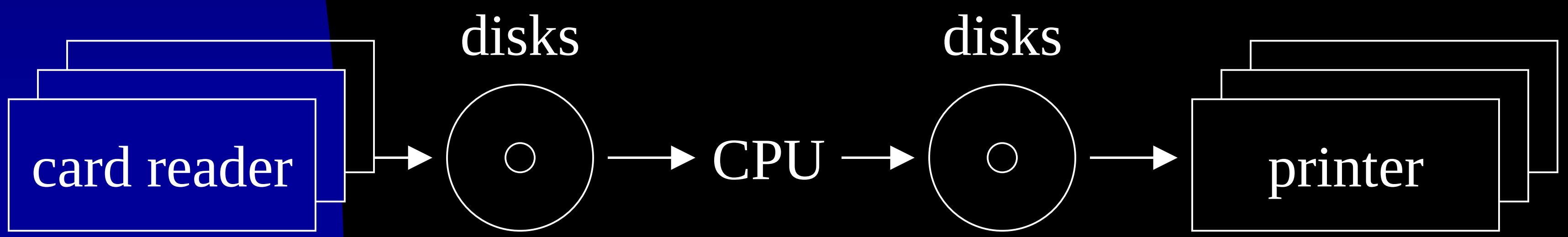
- The Recent Trend:
  - Hyperthreading Processors
  - Multiple Cores over a Single Chip
    - N Standard Processors!
- Loosely-Coupled Systems
  - Processors do not share memory or a clock
  - Blade Servers
    - Each blade-processor board boots independently and runs its own OS.

# Clustered Systems

- Definition: Clustered computers which share storage and are closely linked via LAN networking.
- Advantages: high availability, performance improvement, etc.
- Types
  - Asymmetric/symmetric clustering
  - Parallel clustering – multiple hosts that access the same data on the shared storage.
- Distributed Lock Manager (DLM)
  - Oracle

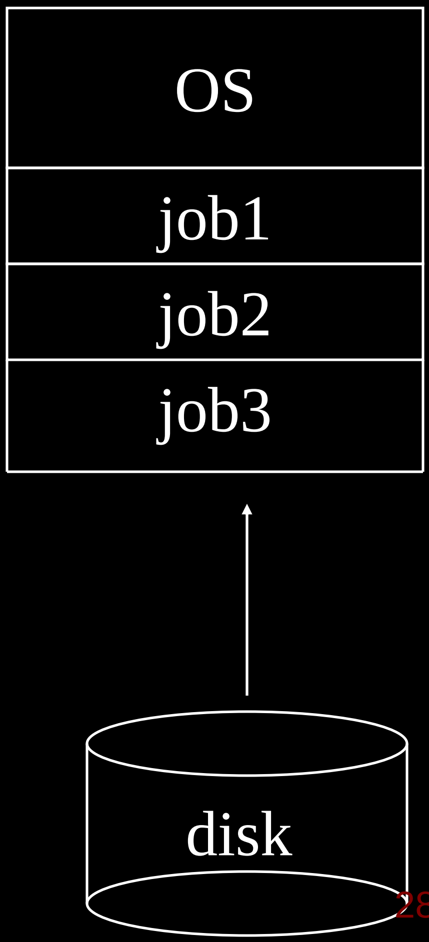
# Operating-System Structure

- Simple batch systems
  - Resident monitor – Automatically transfer control from one job to the next
- Spooling (Simultaneous Peripheral Operation On-Line)
  - Replace sequential-access devices with random-access device



# Operating-System Structure

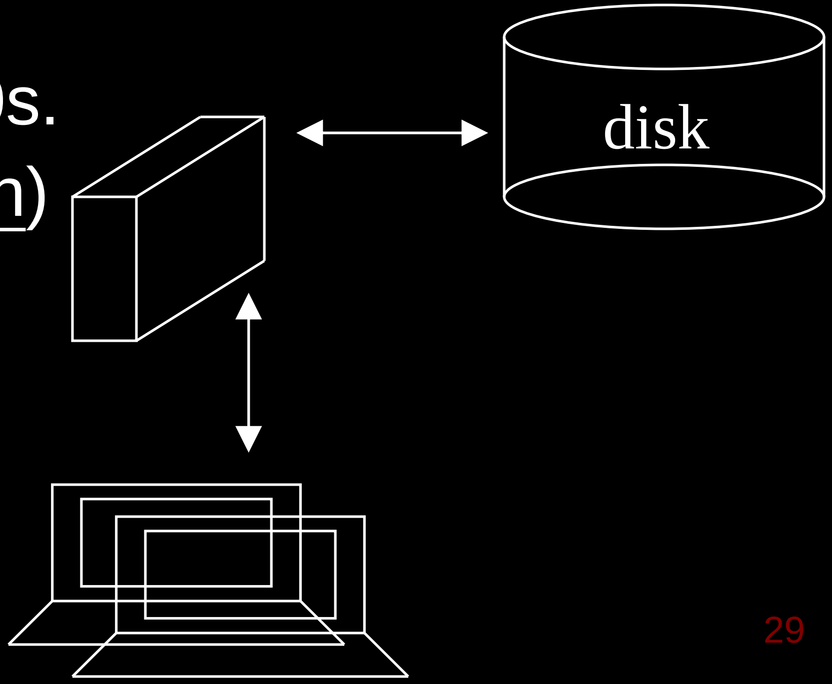
- Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute – Early 1960
  - Job scheduling and CPU scheduling
  - Goal : efficient use of scare resources



# Operating-System Structure

- Time sharing (or multitasking) is a logical extension of multiprogramming!
  - Started in 1960s and become common in 1970s.
  - An interactive (or hand-on) computer system
  - Multics, IBM OS/360
- Virtual Memory
  - Physical Address

on-line file system
virtual memory
sophisticated CPU scheduling
job synchronization
protection & security
……
and so on



# Operating-System Operations

- An Interrupt-Driven Architecture for Modern OS's
  - Events are almost always signaled by the occurrence of an interrupt or a trap (or an exception).
- Protection of User Programs and OS
  - Multiprogramming
  - Sharing of Hardware and Software

# Hardware Protection

- Goal:
  - Prevent errors and misuse!
    - E.g., input errors of a program in a simple batch operating system
    - E.g., the modifications of data and code segments of another process or OS
- Dual-Mode Operations – a mode bit
  - User-mode executions except those after a trap or an interrupt occurs.
  - Monitor-mode (system mode, privileged mode, supervisor mode)
    - Privileged instruction: Machine instructions that may cause harm

# Hardware Protection

- More Modes:
  - One for the Virtual Machine Manager – It provides an interface that is identical to the underlying bare hardware.
  - More for different kernel components

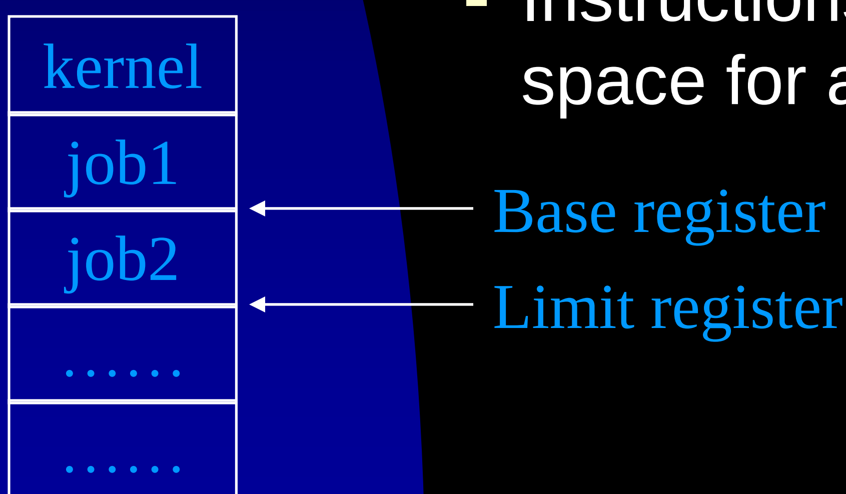
| | | | |
|---|---|---|---|
| virtual user mode | processes | processes | processes |
| virtual monitor mode | kernel 1 | kernel 2 | kernel 3 |
| monitor mode | virtual machine software | | |
| | hardware | | |

# Hardware Protection

- System Calls – trap to OS for executing privileged instructions.
- Resources to protect
  - I/O devices, Memory, CPU
- I/O Protection (I/O devices are scare resources!)
  - I/O instructions are privileged.
    - User programs must issue I/O through OS
    - User programs can never gain control over the computer in the system mode.

# Hardware Protection

- Memory Protection
  - Goal: Prevent a user program from modifying the code or data structures of either the OS or other users!
  - Instructions to modify the memory space for a process are privileged.




⇔ Check for every memory address by hardware

# Hardware Protection

- CPU Protection
  - Goal
    - Prevent user programs from sucking up CPU power!
  - Use a timer to implement time-sharing or to compute the current time.
    - Instructions that modify timers are privileged.
  - Computer control is turned over to OS for every time-slice of time!
    - Terms: time-sharing, context switch

# System Components – Process Management

- Process Management
  - Process: An Active Entity
    - Physical and Logical Resources
      - Memory, I/O buffers, data, etc.
    - Data Structures Representing Current Activities:

Program (code)

\+

Program Counter
Stack
Data Section
CPU Registers
….
And More

# System Components – Process Management

- Services
  - Process creation and deletion
  - Process suspension and resumption
  - Process synchronization
  - Process communication
  - Deadlock handling

# System Components – Memory Management

- Memory: a large array of words or bytes, where each has its own address
- OS must keep several programs in memory to improve CPU utilization and user response time
- Management algorithms depend on the hardware support
- Services
  - Memory usage and availability
  - Decision of memory assignment
  - Memory allocation and deallocation

# System Components – File-System Management

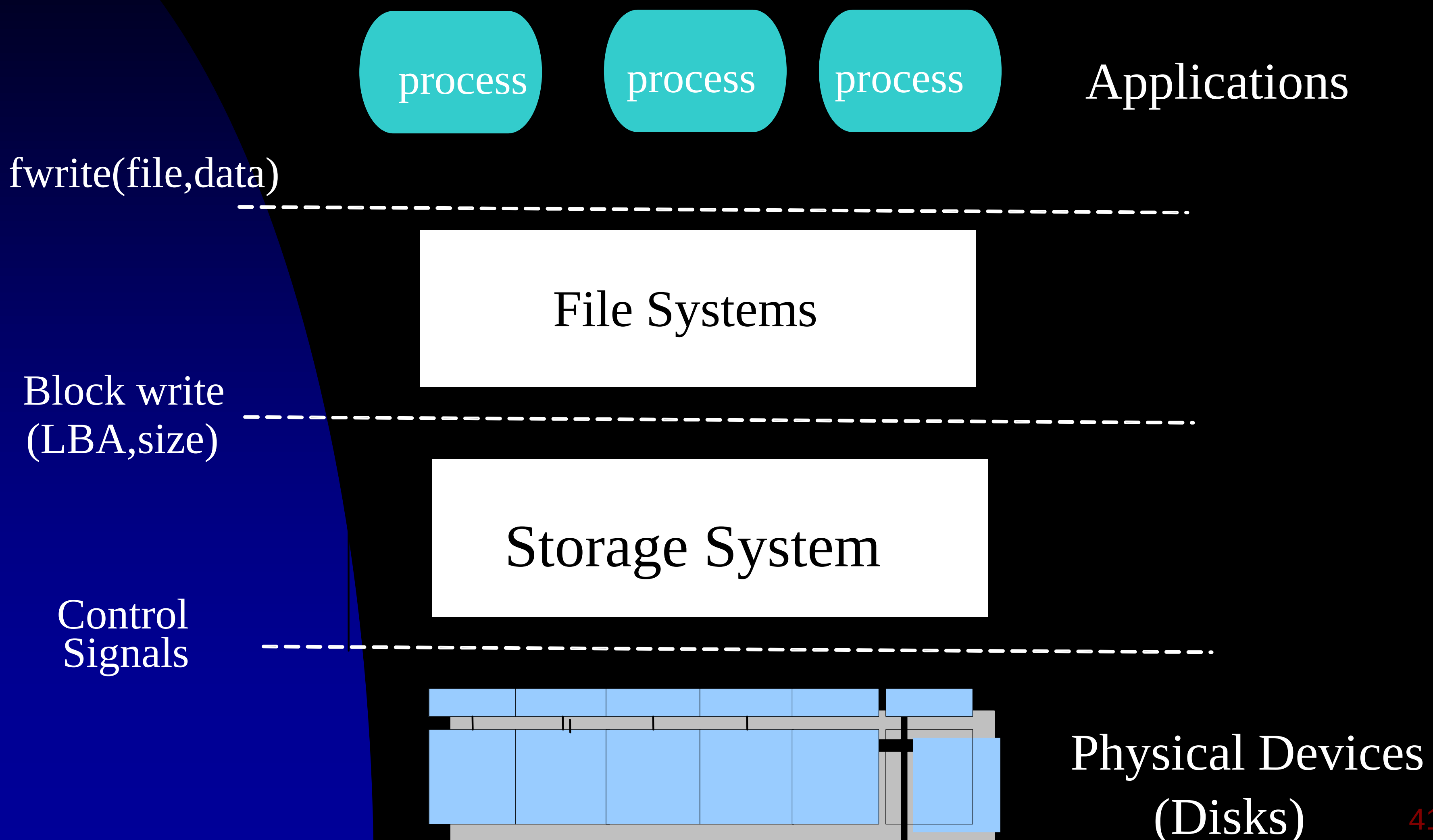
- Goal:
  - A uniform logical view of information storage
  - Each medium controlled by a device
    - Magnetic tapes, magnetic disks, optical disks, etc.
- OS provides a logical storage unit: File
  - Formats:
    - Free form or being formatted rigidly.
  - General Views:
    - A sequence of bits, bytes, lines, records

# System Components – File-System Management

- Services
  - File creation and deletion
  - Directory creation and deletion
  - Primitives for file and directory manipulation
  - Mapping of files onto secondary storage
  - File Backup

* Privileges for file access control

# System Components – File-System Management



# System Components – Mass-Storage Management

- Goal:
  - On-line storage medium for programs & data
    - Backup of main memory
- Services for Disk Management
  - Free-space management
  - Storage allocation, e.g., continuous allocation
  - Disk scheduling, e.g., FCFS

# System Components –Tertiary Storage Devices

- Goals:
  - Backups of disk data, seldom-used data, and long-term archival storage
- Examples:
  - Magnetic tape drives and their tapes, CD & DVD drives and platters.
- Services – OS Supports or Applications' Duty
  - Device mounting and unmounting
  - Exclusive allocation and freeing
  - Data transfers from tertiary devices to secondary storage devices.

# System Components – I/O System Management

- Goal:
  - Hide the peculiarities of specific hardware devices from users
- Components of an I/O System
  - A buffering, caching, and spooling system
  - A general device-driver interface
  - Drivers

# Caching

Speed

register

Cache

<--->

High hitting rate

• instruction & data cache

• combined cache

Main Memory

Volatile Storage

Electronic Disk

Faster than magnetic disk – nonvolatile?!
Alias: RAM Disks

Cost

Magnetic Disk

Optical Disk

Magnetic Tape

Sequential Access

*XX* GB/350F

# Caching

| Level | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Name | Registers | Cache | Memory | Disk |
| Typical Size | < 1KB | > 16MB | > 16GB | > 100GB |
| Implementation Strategy | Custom memory with multiple ports, CMOS | On-chip or off-chip CMOS SRAM | CMOS DRAM | Magnetic Disks |
| Access Time (ns) | 0.25 – 0.5 | 0.5 – 2.5 | 80 – 250 | 5,000,000 |
| Bandwidth (MB/s) | 20,000 – 100,000 | 5000 – 10,000 | 1000 – 5000 | 20 – 150 |
| Managed by | Compiler | Hardware | OS | OS |
| Backup by | Cache | Memory | Disk | CD/Tape |

# Caching

- Caching
  - Information is copied to a faster storage system on a temporary basis
  - Assumption: Data will be used again soon.
    - Programmable registers, instr. cache, etc.
- Cache Management
  - Cache Size and the Replacement Policy
- Movement of Information Between Hierarchy
  - Hardware Design & Controlling Operating Systems

# Caching

- Coherency and Consistency
  - Among several storage levels (vertical)
    - Multitasking vs unitasking
  - Among units of the same storage level , (horizontal), e.g. cache coherency
    - Multiprocessor or distributed systems

# Protection and Security

- Goal
  - Resources are only allowed to be accessed by authorized processes.
- Definitions:
  - Protection – any mechanism for controlling the access of processes or users to the resources defined by the computer system.
  - Security – Defense of a system from external and internal attacks, e.g., viruses, denial of services, etc.

# Protection and Security

- Protected Resources
  - Files, CPU, memory space, etc.
- Protection Services
  - Detection & controlling mechanisms
  - Specification mechanisms
- Distinguishing of Users
  - User names and ID's
  - Group names and GID's
  - Privilege Escalating, e.g., Setuid in Unix
    - To gain extra permissions for an activity.
- Remark: Reliability!

# Kernel Data Structures

- Frequently Used Data Structures
  - Array, List, Stack, Queue, Tree, Hash
  - Bitmaps – A string of *n* binary digits to represent the status of *n* items.
    - Advantage: Space Efficiency
    - An example is the availability status of disk blocks.

# Computing Environments

- Evolving Environments
  - Transition from the period of scarce resources to the period of ubiquitous access!
  - In the past, portability is achieved by laptops!
    - Remote access is supported in a limited way. Mainframes are prevalent!
  - Now, PC's, mobile devices, and various equipments are connected!
    - High speed networks are available at home and office! Web-computing is popular (e.g., portals).

# Computing Environments

- Mobile Computing
  - Trends: Computing on handheld smartphones and tablets now offers tremendous growth in the wide range of applications, such as email and GPS, augmented-reality applications, but with limitation on screen size, memory/storage capacity, and power/energy consumption.

# Computing Environments

- Distributed/Loosely-Coupled Systems: Heterogeneous or homogeneous computer systems that are networked to provide access to various resources
  - Depend on networking for their functionality
    - Networks vary by the protocols used: TCP/IP, ATM, etc.
  - Be characterized by their node distances
    - Local-area network (LAN)
    - Wide-area network (WAN)
    - Metropolitan-area network (MAN)
    - Personal-area network – distance of few feet

# Computing Environments

  - Media – copper wires, fiber strands, satellite wireless transmission, infrared communication,etc.
- Network Operating Systems
  - Autonomous computers
  - A distributed operating system – a single OS controlling the network.

# Computing Environments

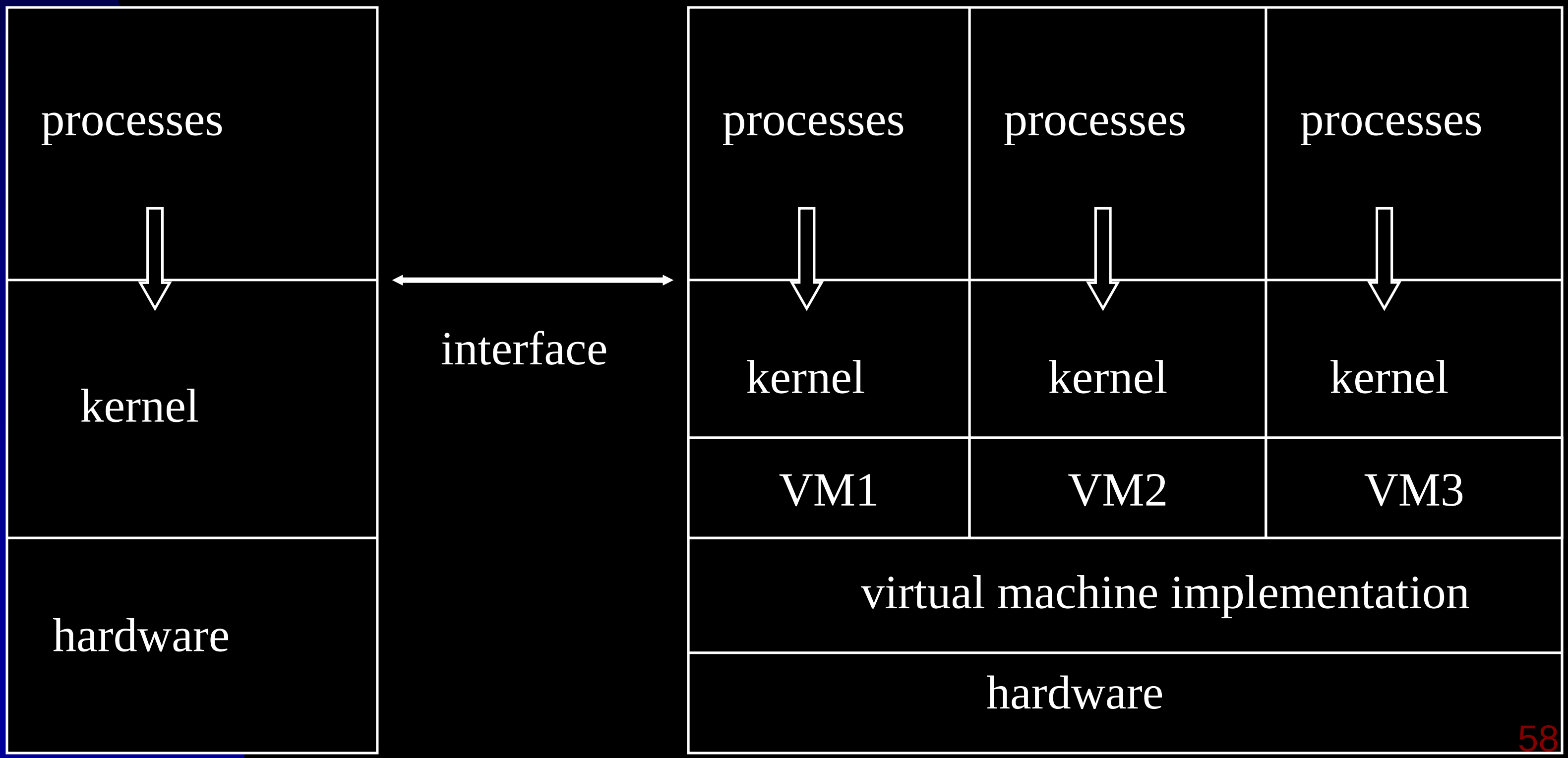
- Peer-to-Peer Systems
  - Characteristics: Client and server roles depend on who is requesting or providing a service.
  - Network connectivity is an essential component.
  - Service Availability and Discovery
    - Registration of services: a centralized lookup service or not
    - A discovery protocol
  - Issues:
    - Legal problems in exchanging files.

# Computing Environments

- Client-Server Systems
  - Trend: The functionality of clients is improved in the past decades.
  - Categories:
    - Compute-server systems
    - File-server systems

# Computing Environments

- Virtual Machines: provide an interface that is identical to the underlying bare hardware

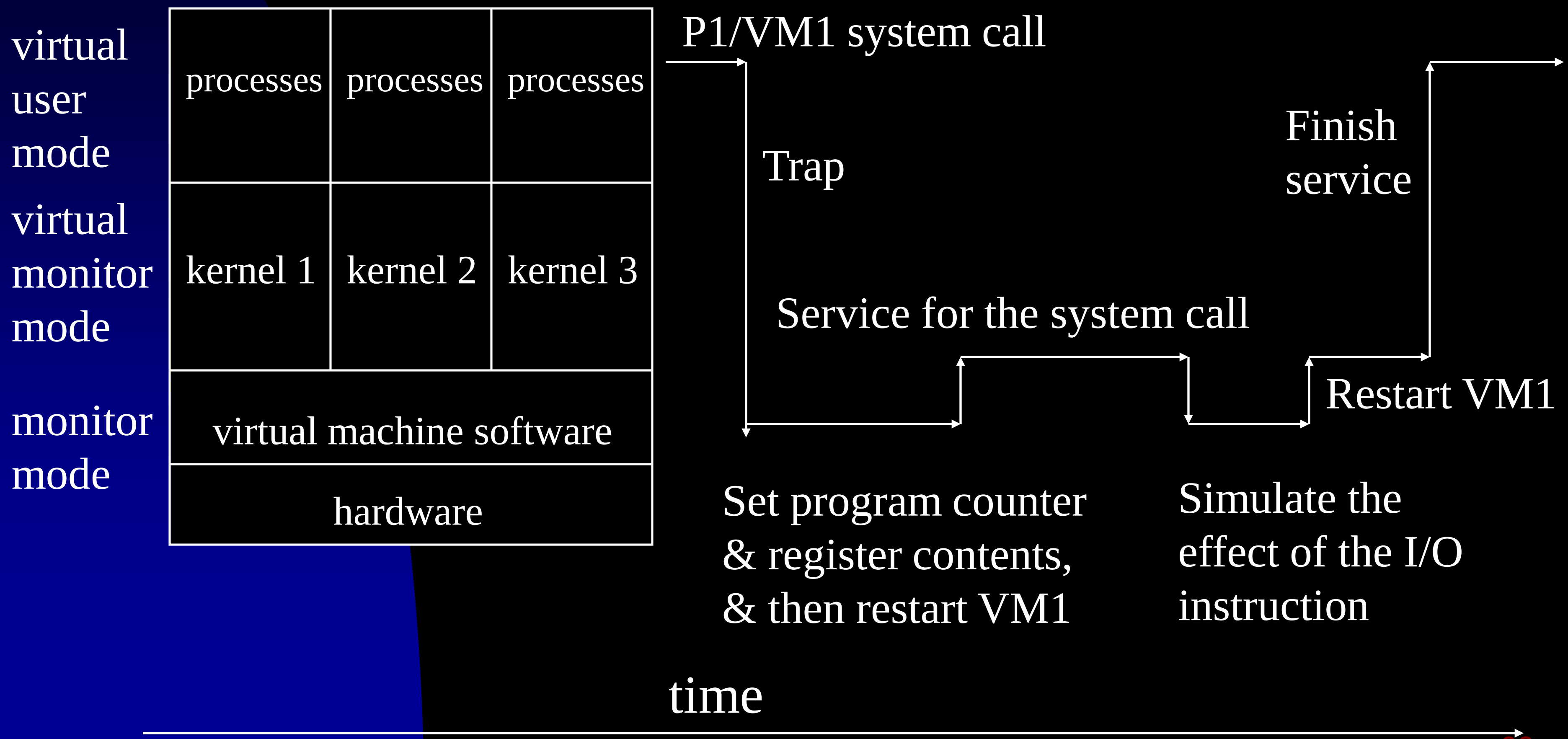


# Computing Environments

- Implementation Issues of Virtual Machines:
  - Emulation of Physical Devices
    - E.g., Disk Systems
      - An IBM minidisk approach
  - User/Monitor Modes
    - (Physical) Monitor Mode
      - Virtual machine software
    - (Physical) User Mode
      - Virtual monitor mode & Virtual user mode

# Computing Environments

- How a Virtual Machine works:



# Computing Environments

- Disadvantages of Virtual Machines:
  - Slow!
    - Execute most instructions directly on the hardware
    - Emulation is slow but is needed for obsolete hardware.
  - No direct sharing of resources
    - Physical devices and communications

* I/O could be slow (interpreted) or fast (spooling)

# Computing Environments

- Advantages of Virtual Machines:
  - Complete Protection – Complete Isolation！
  - OS Research & Development
    - System Development Time
  - Extensions to Multiple Personalities, such as Mach (software emulation)
    - Emulations of Machines and OS's, e.g., Windows over Linux
  - System Consolidation

* Simulation:  Programs of a guest system are run on an emulator that translate each  of the guest system instructions into the native instruction set of the host system.

# Computing Environments

- Cloud Computing - Delivers computing, storage, and even applications as a service across a network
- Types
  - Public, Private and Hybrid Clouds
  - Software as a service (SaaS), e.g., Gmail.
  - Platform as a service (PaaS), e.g., database server.
  - Infrastructure as a service (IaaS), e.g., storage for backup.

# Computing Environments

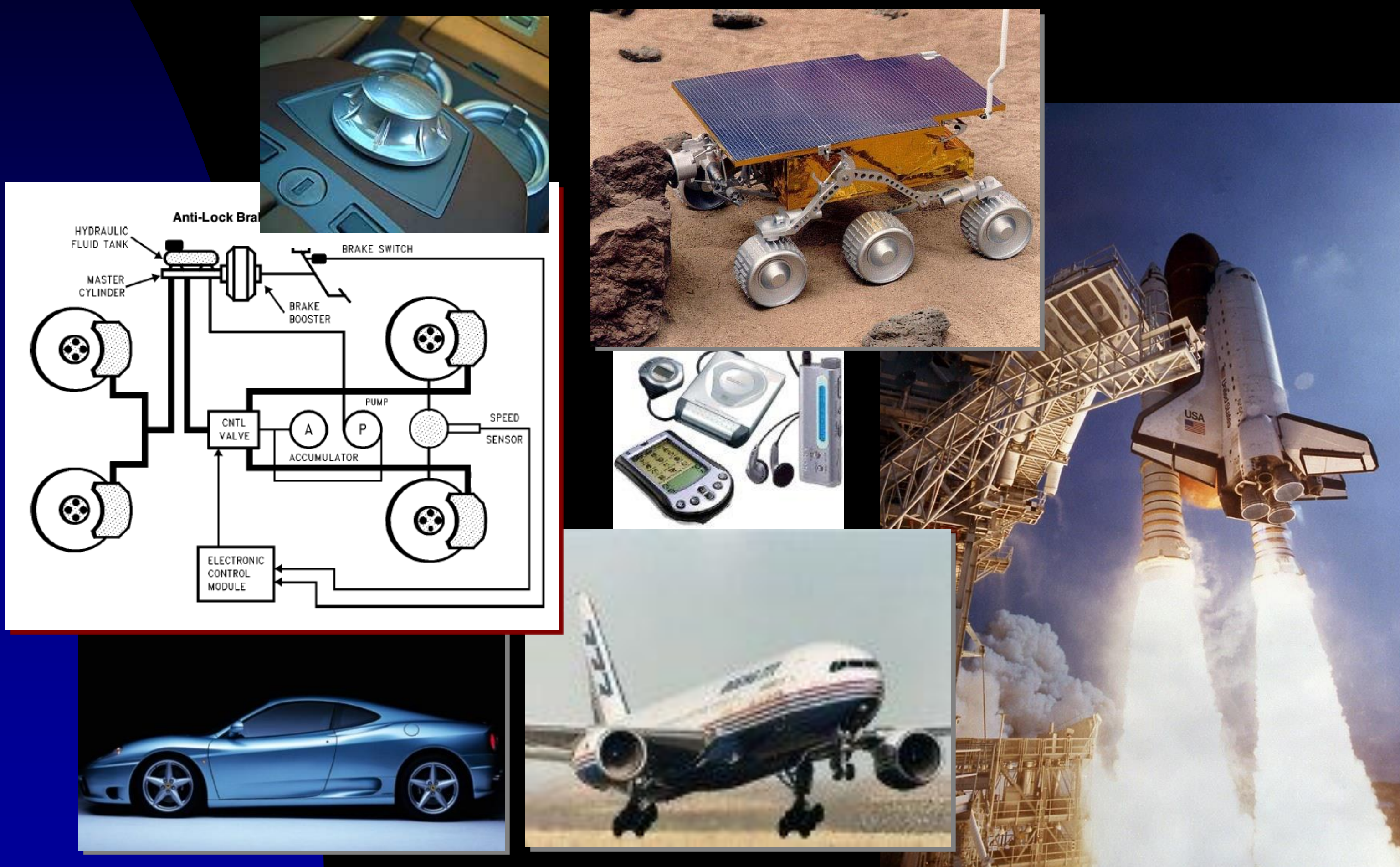
- Web-Based Computing
  - Web Technology
    - Portals, network computers, etc.
  - Network connectivity
  - New categories of devices
    - Load balancers
- Embedded Computing
  - Car engines, robots, VCR's, home automation
  - Embedded OS's often have limited features.

# Computing Environments

- Embedded Computers – Most Prevalent Form of Computers
  - Have a wide variety ranged from car engines to VCR's.
    - General-purpose computers with standard OS's, HW devices with or without embedded OS's
    - Standalone units or members of networks and the Web
  - Tend to have specific tasks and almost always run real-time operating systems.

# Computing Environments

- Real-Time Embedded Computers



# Computing Environments

- Definition: A real-time system is a computer system where a timely response by the computer to external stimuli is vital!
- Hard real-time system: The system has failed if a timing constraint, e.g. deadline, is not met.
  - All delays in the system must be bounded.
  - Many advanced features are absent.

# Computing Environments

- Soft real-time system: Missing a timing constraint is serious but does not necessarily result in a failure unless it is excessive!
  - A critical task has a higher priority.
  - Supported in most commercial OS.
- <u>Real-time</u> means <u>on-time</u> instead of fast

# Open-Source Operating Systems

- Definitions: OS with available source code.
  - Closed-source OS, e.g., MS Widows, or hybrid OS, e.g., iOS.
  - Arguably issues on bugs, security, support, etc.
  - Examples: GNU/Linux, BSD UNIX, and Solaris (up to 2005 versions).