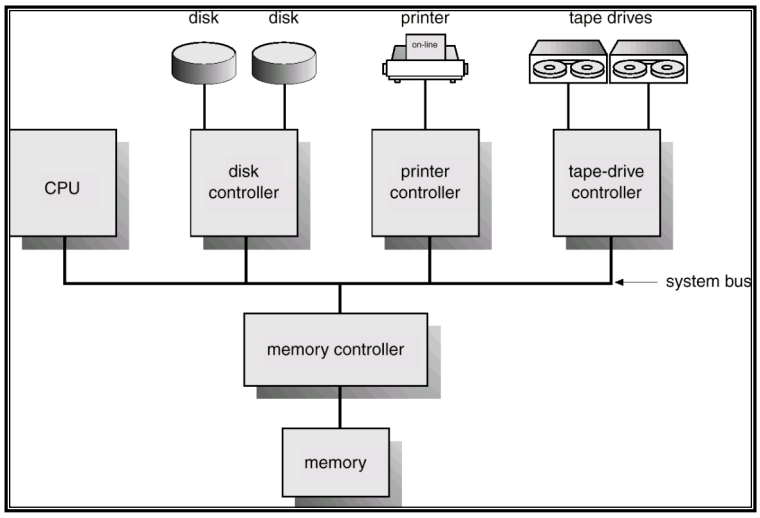
CMSC 412
Fall 2004

Computer and Operating System Structures

Announcements

- Project #0
  - Due Friday
  - Submission program is now working
- Project #1
  - Much harder than Project #0!
  - Posted tomorrow
- Reading
  - Chapter 2,3
  - Chapter 3,4 (for Monday)

Computer Systems



I/O Subsystem

- Many different types of devices
  - disks
  - networks
  - displays
  - mouse
  - keyboard
- Each has different peak performance
  - bandwidth
    - rate at which data can be moved
  - latency
    - time from request to first data back

Impacts System Structure

- Different Buses (not just one!)
  - Processor Bus (on chip)
    - Many Gigabytes/sec
  - Memory Bus (on processor board)
    - ~1-2 Gigabytes/sec
  - I/O Bus (PCI, MCA)
    - ~100 Megabytes/sec
  - Device Bus (SCSI, USB)
    - tens of Megabytes/sec

Impacts System Structure

- Different ways to transmit data
  - Interrupt-driven I/O
    - CPU directs how data is transmitted to and from the device
    - User interrupts CPU to tell data to be transmitted
    - Device interrupts CPU when task is completed or data is available
  - Direct Memory Access (DMA)
    - CPU directs transmission at a much more coarse-grained level

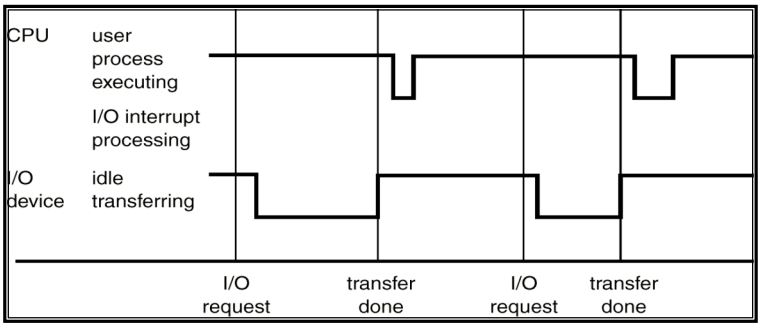
I/O Interrupts

- Indicate an event has occurred
  - can be caused by hardware devices
    - indicates data present or hardware available
  - can be caused by software
    - system call (or *trap*)
- Interrupt the CPU to execute a *handler*
  - saves state about what was happening
  - returns where it left off when finished

Servicing Interrupts

- Need to know what device interrupted
  - could ask each device (slow!)
  - instead use an interrupt vector
    - array of pointers to functions to handle a specific interrupt
- What happens if an interrupt arrives while we are in an interrupt handler?
  - OS may wish to *disable interrupts* until handling is complete.

Servicing Interrupts



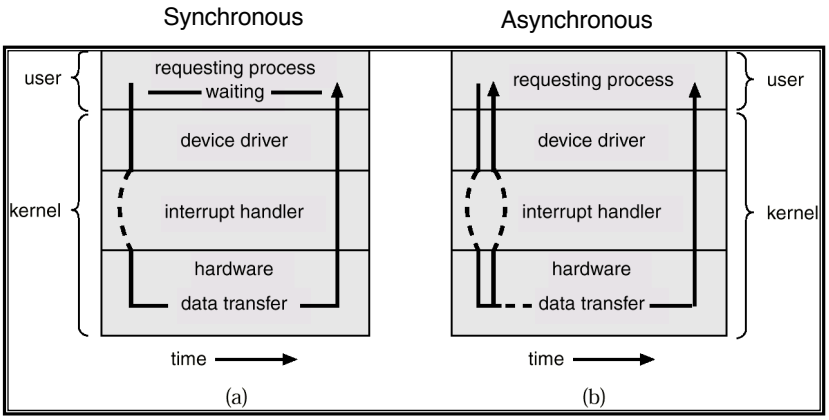
Synchronous I/O

- After I/O starts, control returns to user program only upon I/O completion.
  - *Wait* instruction or loop idles the CPU until the next interrupt
  - At most one I/O request is outstanding at a time; no simultaneous I/O processing.
- Modern OSs do not perform synchronous I/O
  - Not compatible with multitasking
  - May provide the illusion of synchronous I/O to user programs, however

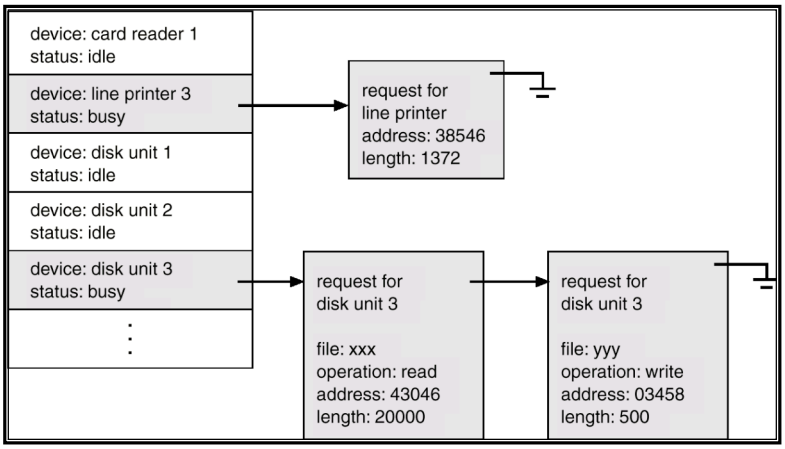
Asynchronous I/O

- After I/O starts, control returns to (some) user program without waiting 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 process interrupt.

Two I/O Methods



Device-status Table



Direct Memory Access (DMA)

- Used for high-speed I/O devices able to transmit information at close to memory speeds.
- One interrupt is generated per *block*, rather than the one interrupt per *byte*.
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.

System Calls

- A software-generated interrupt
  - a.k.a. *trap*
- Provide the interface between application programs and the OS kernel
- Are like procedure calls
  - take parameters
  - calling routine waits for response

System Call Mechanism

- Use numbers to indicate what call is made
- Parameters stored in registers or the stack
- Why do we use system call numbers rather than directly calling a kernel subroutine?
  - permits changing the size and location of system call implementations without having to re-link application programs

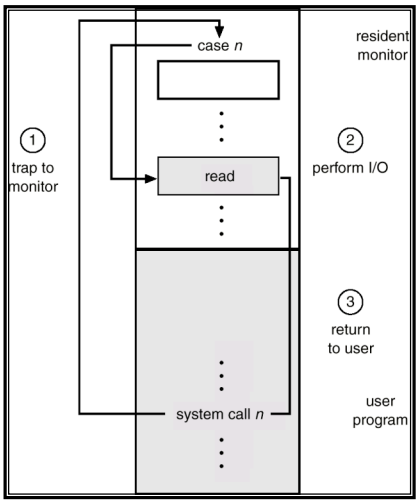
GeekOS and x86

- Intel system call instruction
  - **int** *n* where *n* is the interrupt vector #
  - "call kernel routine *n*"
  - vectors 0-31 reserved
    - Page fault, segmentation violation, etc.
- GeekOS
  - All system calls set *n* as 90
  - System call number stored in *eax* register

Types of System Calls

- File Related
  - open, create, read, write, close, delete
  - get or set file attributes
- Informational
  - get time
  - set system data (OS parameters)
  - get process information (id, time used)
- Communication Related
  - establish a connection; terminate a connection
  - send, receive messages
- Process control
  - create/terminate a process (including self)

Use of a System Call for I/O



Why use system calls at all?

- Why not “link” application programs against the kernel and call kernel routines directly?

Protection

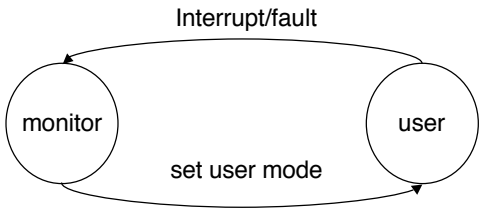
- Need to protect OS from user programs and user programs from each other
  - Don't want a bug in a user program to crash the whole machine (as in earlier OSs, like MS-DOS, MacOS, Windows 3.1, and others)
- Hardware resources of interest
  - Memory, I/O devices, CPU

Dual-Mode Operation

- Provide hardware support to differentiate between at least two modes of operations.
  1. *User mode* – execution for a user.
  2. *Monitor mode* (also *kernel mode* or *system mode*) – execution done on behalf of operating system.
- Operations in user mode a subset of those allowed in monitor mode
  - Privileged instructions only in monitor

Dual-Mode Operation

- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1). X86 actually has 4 modes (2 bits).
- When an interrupt or fault occurs hardware switches to monitor mode.



I/O Protection

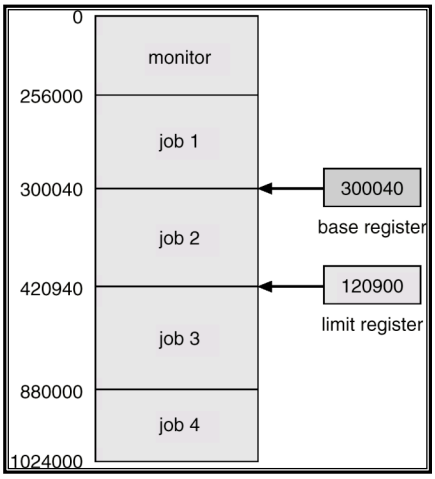
- System call mechanism prevents user mode programs from accessing devices directly.
  - All I/O instructions are privileged
- But what if user program can overwrite interrupt handler with its own code?

12

Memory Protection

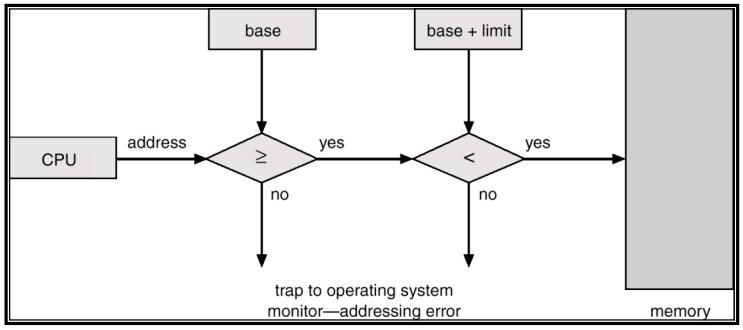
- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - **Base register** – holds the smallest legal physical memory address.
  - **Limit register** – contains the size of the range
- Memory outside the defined range is protected.

Base and Limit Registers



13

Hardware Address Protection



Changing the base and limit registers are privileged operations

CPU Protection

- *Timer* – interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
- Commonly used to implement time sharing.
- Load-timer is a privileged instruction.

14

Storage Structure

- Main memory – only large storage media that the CPU can access directly.
- Secondary storage – extension of main memory that provides large *nonvolatile* storage capacity.

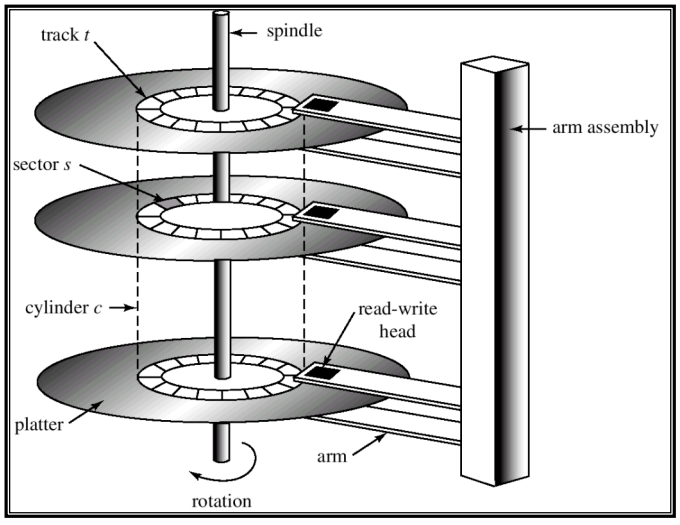
Disks

- Several types:
  - Hard Disks - rigid surface with magnetic coating
  - Floppy disks - flexible surface with magnetic coating
  - Optical (CDs and DVDs) - read only, write once, multi-write
- The *disk controller* determines the logical interaction between the device and the computer.

Hard Disks

- Collection of platters
- Platters contain concentric rings called tracks
- Tracks are divided into fixed sized units called sectors
- A cylinder is a collection of all tracks equidistant from the center of disk
- Current Performance:
  - capacity: megabytes to hundreds of gigabytes
  - throughput: sustained < 10 megabytes/sec
  - latency: milliseconds

Moving-Head Disk Mechanism



16

Relevant Disk Ops

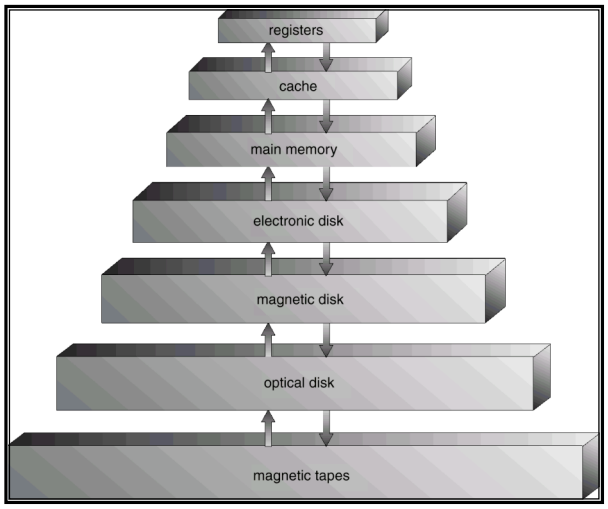
- Seek
  - Move arm to appropriate cylinder
- Read/Write
  - One sector at a time
  - Must wait until disk rotates to appropriate sector address
- OS must keep track of raw disk addresses to implement filesystems

Storage Hierarchy

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility
- *Caching* – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.

17

Storage-Device Hierarchy

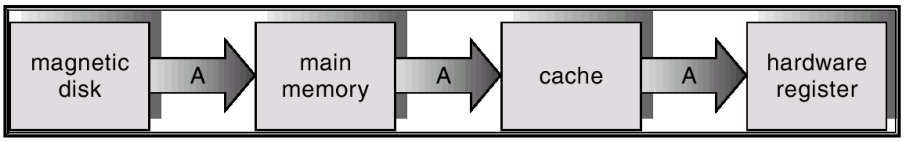


Caching

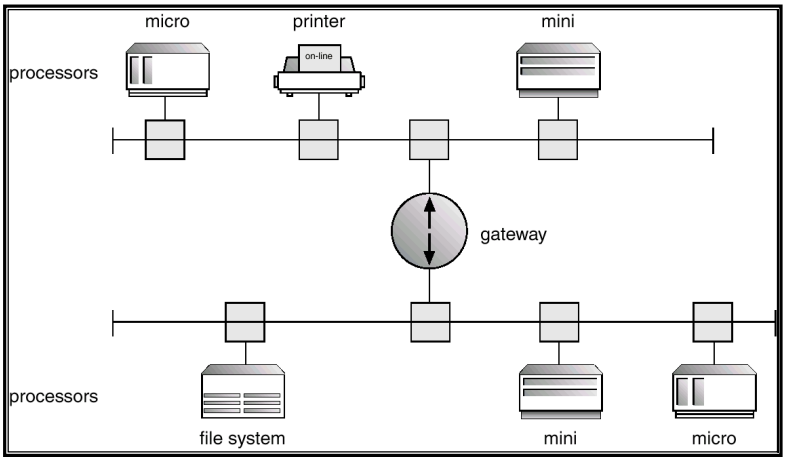
- Use of high-speed memory to hold recently-accessed data.
- Requires a *cache management* policy.
- Caching introduces another level in storage hierarchy. This requires data that is simultaneously stored in more than one level to be *consistent*.

18

Migration of A From Disk to Register

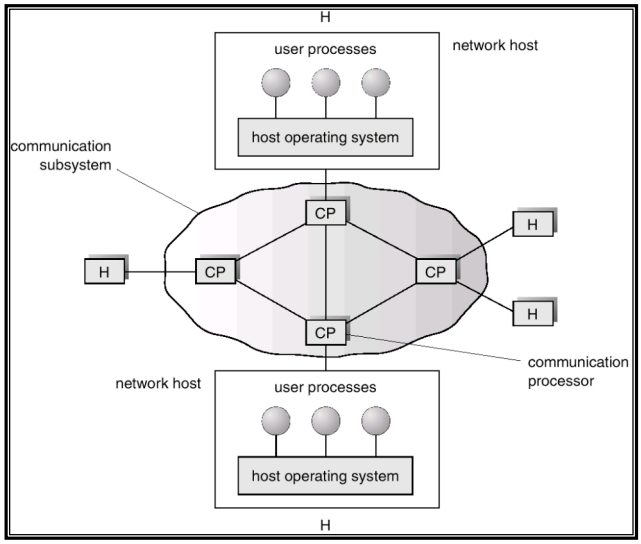


Local Area Network Structure



19

Wide Area Network Structure



20