Operating Systems

Input/Output Devices
(Ch12.1 - 12.3, 12.7; 13.1-13.3, 13.7)

Introduction

- One OS function is to control devices
  - significant fraction of code (80-90% of Linux)
- Want all devices to be simple to use
  - convenient
  - ex: stdin/stdout, pipe, re-direct
- Want to optimize access to device
  - efficient
  - devices have very different needs

Outline

- Introduction ✓
- Hardware ¬
- Software
- Specific Devices
  - Hard disk drives
  - Clocks
  - Terminals

Hardware

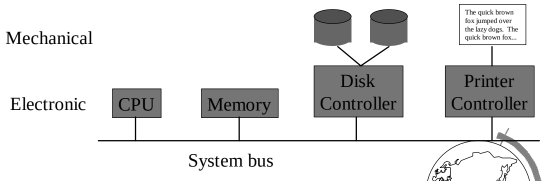
- Types of I/O devices
- Device controllers
- Direct Memory Access (DMA)

I/O Device Types

- block - access is independent
  - ex- disk
- character - access is serial
  - ex- printer, network
- other
  - ex- clocks (just generate interrupts)

Device Controllers

- Mechanical and electronic component



- OS deals with electronic
  - device controller

Direct Memory Access (DMA)

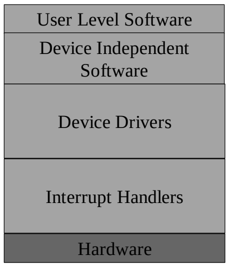
- ✦ Very Old
  - – Controller reads from device
  - – OS polls controller for data
- ✦ Old
  - – Controller reads from device
  - – Controller interrupts OS
  - – OS copies data to memory
- ✦ DMA
  - – Controller reads from device
  - – Controller copies data to memory
  - – Controller interrupts OS

Outline

- ✦ Introduction ✓
- ✦ Hardware ✓
- ✦ Software ¬
- ✦ Specific Devices
  - – Hard disk drives
  - – Clocks
  - – Terminals

I/O Software Structure

- ✦ Layered



(Talk from bottom up)

Interrupt Handlers

CPU

1) Device driver initiates I/O

(CPU executing, checking for interrupts between instructions)

3) Receives interrupt, transfer to handler

4) Handler processes

(*Resume processing)*

I/O Controller

1) Initiates I/O

(I/O device processing request)

2) I/O complete. Generate interrupt.

Interrupt Handler

- ✦ Make interrupt handler as small as possible
  - – interrupts disabled
- ✦ Do minimal amount of work
  - – defer rest until later in the rest of the device driver
  - – deferred procedure call
- ✦ Implementation specific
  - – 3rd party vendors

Device Drivers

- ✦ Device dependent code
  - – includes interrupt handler
- ✦ Accept abstract requests
  - – ex: “read block n”
- ✦ See that they are executed by device hardware
  - – registers
  - – hardware commands
- ✦ After error check
  - – pass data to device-independent software

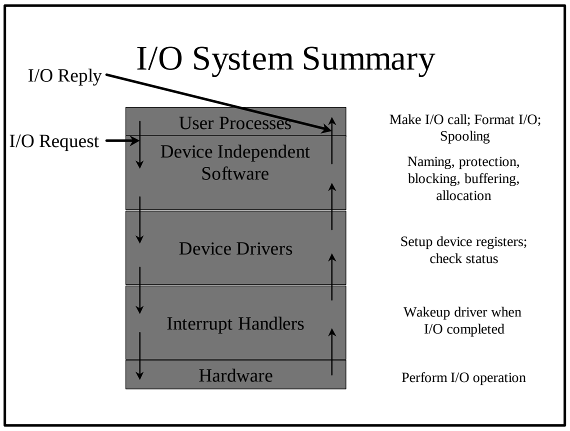
Device-Independent I/O Software

- Much driver code independent of device
- Exact boundary is system-dependent
  - sometimes inside for efficiency
- Perform I/O functions common to all devices
- Examples:
  - naming protection block size
  - buffering storage allocation error reporting

User-Space I/O Software

- Ex: `count = write( fd, buffer, bytes);`
- Put parameters in place for system call
- Can do more: formatting
  - `printf(), gets()`
- Spooling
  - spool directory, daemon
  - ex: printing, USENET

I/O System Summary



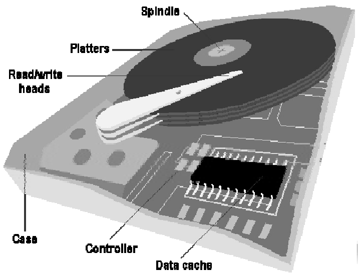
I/O Reply
I/O Request
User Processes
Device Independent Software
Device Drivers
Interrupt Handlers
Hardware
Make I/O call; Format I/O; Spooling
Naming, protection, blocking, buffering, allocation
Setup device registers; check status
Wakeup driver when I/O completed
Perform I/O operation

Outline

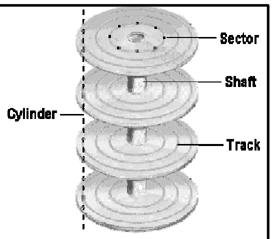
- Introduction ✓
- Hardware ✓
- Software ✓
- Specific Devices ¬
  - Hard disk drives
  - Clocks
  - Terminals

Hard Disk Drives (HDD)

- Controller often on disk
- Cache to speed access



HDD - Zoom



- Platters
  - 3000-10,000 RPM

  (floppy 360 RPM)
- Tracks
- Cylinders
- Sectors

Ex: `hdb: Conner Peripherals 540MB CFS540A, 516MB w/64kB Cache, CHS=1050/16/63`

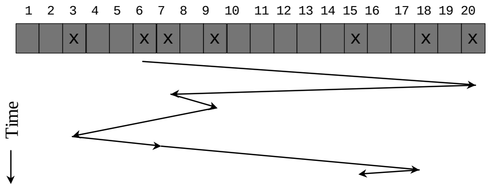
- 1050 cylinders (tracks), 16 heads (8 platters), 63 sectors per track

- Disk Arms all move together
- If multiple drives
  - overlapping seeks but one read/write at a time

Disk Arm Scheduling

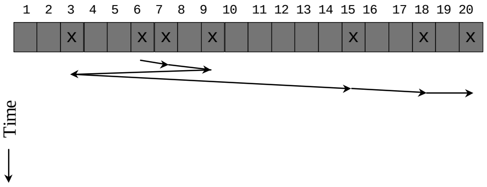
- ✦ Read time:
  - – seek time (arm to cylinder)
  - – rotational delay (time for sector under head)
  - – transfer time (takes bits off disk)
- ✦ Seek time dominates
- ✦ How does disk arm scheduling affect seek?

First-Come First-Served (FCFS)



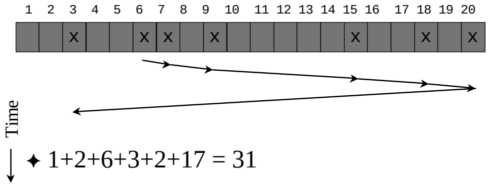
- ✦ 14+13+2+6+3+12+3=53
- ✦ Service requests in order that they arrive
- ✦ Little can be done to optimize
- ✦ What if many requests?

Shortest Seek First (SSF)



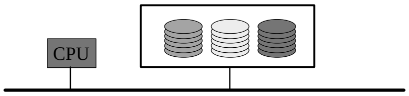
- ✦ 1+2+6+9+3+2 = 23
- ✦ Suppose many requests?
  - – Stay in middle
  - – Starvation!

Elevator (SCAN)



- ✦ 1+2+6+3+2+17 = 31
- ✦ Usually, a little worse than SSF
- ✦ C-SCAN has less variance
- ✦ Note, seek getting faster, rotational not
  - – Someday, change algorithms

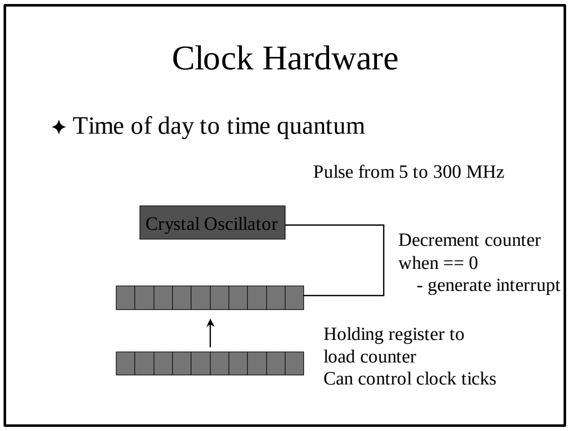
Redundant Array of Inexpensive Disks (RAID)



- ✦ Pull data in parallel
- ✦ For speed
- ✦ For fault-tolerance
  - – Example: 38 disks
  - – Form 32 bit word, 6 check bits

Error Handling

- ✦ Common errors:
  - – programming error (non-existent sector)
  - – transient checksum error (dust on head)
  - – permanent checksum error (bad block)
  - – seek error (arm went to wrong cylinder)
  - – controller error (controller refuses command)



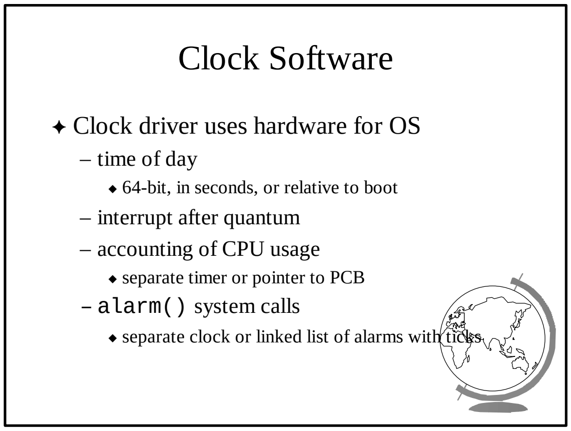
Clock Hardware

- ✦ Time of day to time quantum

Pulse from 5 to 300 MHz

Decrement counter
when == 0
- generate interrupt

Holding register to
load counter
Can control clock ticks



Clock Software

- ✦ Clock driver uses hardware for OS
  - – time of day
    - ◆ 64-bit, in seconds, or relative to boot
  - – interrupt after quantum
  - – accounting of CPU usage
    - ◆ separate timer or pointer to PCB
  - – `alarm()` system calls
    - ◆ separate clock or linked list of alarms with ticks