Operating Systems

Input/Output Devices
(Ch 5: 5.1 -5.5)

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 (done)
• Hardware
• Software
• Specific Devices
  – Hard disk drives
  – Clocks

Hardware

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

Device Controllers

• Mechanical and electronic component
  • Mechanical
  • Electronic
• OS deals with electronic
  – device controller

I/O Device Types

• block - access is independent
  – ex- disk
• character - access is serial
  – ex- printer, network
• other
  – ex- clocks (just generate interrupts)
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 (done)
- Hardware (done)
- Software
  - Specific Devices
    - Hard disk drives
    - Clocks
    - Terminals

I/O Software Structure
- Layered
  - User Level Software
  - Device Independent Software
  - Device Drivers
  - Interrupt Handlers
  - Hardware

(Talk from bottom up)

Interrupt Handlers
- CPU
  1) Device driver initiates I/O
  (CPU executing, checking for interrupts between instructions)
  2) I/O complete. Generate interrupt.
- I/O Controller
  1) Initiates I/O (I/O device processing request)
  2) I/O complete. Generate interrupt.
  3) Receives interrupt, transfer to handler
  4) Handler processes (Resume processing)

Interrupt Handler
- Make interrupt handler as small as possible
  - interrupts disabled
  - Split into two pieces
- First part does minimal amount of work
  - defer rest until later in the rest of the device driver
  - Windows: “deferred procedure call” (DPC)
  - Linux: “top-half” handler
- Second part does most of work
- 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
- User Processes
  - Make I/O call, Format I/O
  - Spooling
    - Naming, protection, blocking, buffering, allocation
  - Setup device registers; check status
  - Wakeup driver when I/O completed
- Device Independent Software
  - Perform I/O operation
- Device Drivers
- Interrupt Handlers
- Hardware

Outline
- Introduction (done)
- Hardware (done)
- Software (done)
- Specific Devices
  - Hard disk drives
  - Clocks

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: Indi: Conner Peripherals 540MB
    - CF5540A, 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 (take bits off disk)
- Seek time dominates
- How does disk arm scheduling affect?

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 avg seek time than SSF
  - But avoids more fair, avoids starvation
- C-SCAN has less variance
- Note, seek getting faster, rotational not
  - Someday, change algorithms

Redundant Array of Inexpensive Disks (RAID)

- For speed
  - Pull data in parallel
- For fault-tolerance
  - Example: 38 disks, form 32 bit word, 6 check bits
  - Example: 2 disks, have exact copy on one disk

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

  - Crystal Oscillator
  - Decrement counter when == 0
    - generate interrupt
  - Holding register to load counter
  - Can control clock ticks

Clock Software Uses

- 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