Input/Output Devices

The Device Driver Interface

Device Management Organization

System Call Interface

Example: BSD UNIX Driver

Overlapping the Operation of a Device and the CPU
Overlapping CPU-Controller Operations in a Process

Polling I/O Read Operation

Interrupt-driven I/O Operation

Device Independent Function Call

Driver-Kernel Interface

Drivers are distinct from main part of kernel
Kernel makes calls on specific functions, drivers implement them
Drivers use kernel functions for:
- Device allocation
- Resource (e.g., memory) allocation
- Scheduling
- etc. (varies from OS to OS)
Reconfigurable Device Drivers

System call interface

Entry Points for Device j

Driver for Device j

Other Kernel services

Handling Interrupts

Device driver j

int read(...) {
  // Prepare for I/O
  save_state(j);
  out dev;
  // Done (no return)
}

Device interrupt handler j

void dev_handler(...) {
  // Cleanup after op
  get_state(j);
  signal(dev[j]);
  return_from_sys_call();
}

Device status table

Device Controller

Interrupt Handler

Handling Interrupts(2)

Device driver j

int read(...) {
  // Prepare for I/O
  save_state(j);
  out dev;
  return_from_sys_call();
  wait(dev[j]);
}

Device interrupt handler j

void dev_handler(...) {
  // Cleanup after op
  signal(dev[j]);
  return_from_sys_call();
}

Device status table

Device Controller

Interrupt Handler

The Pure Cycle Water Company

Customer Office

Water Company

Returning the Empties

Water Consumers

Delivering Water

Hardware Buffering

Process Controller

Data

Device

Unbuffered

Process reads b_i - Controller reads b_i -

Process reads b_i - Controller reads b_i -

Double Buffering in the Driver

Process Controller

A B

Device

Process Controller

A B

Device

Driver

Hardware

Controller

A B

Device

Controller

A B

Device
Circular Buffering

From data producer

To data consumer

A Generic Communications Device

Bus

Generic Controller

Communications Controller

Local Device

Device

Cabling connecting the controller to the device

• Printer
• Modem
• Network

Rotating Media

(a) Multi-surface Disk
(b) Disk Surface
(b) Cylinders

Storage Device

Device Driver API

SCSI API

• commands
• bits per byte
• etc.

Controller

Magnetic Disk

Magnetic Disk

Compute vs I/O Bound

Compute-bound

I/O-bound

Disk Optimizations

• Transfer Time: Time to copy bits from disk surface to memory
• Disk latency time: Rotational delay waiting for proper sector to rotate under R/W head
• Disk seek time: Delay while R/W head moves to the destination track/cylinder
• Access Time = seek + latency + transfer
Optimizing Seek Time

- Multiprogramming on I/O-bound programs => set of processes waiting for disk
- Seek time dominates access time => minimize seek time across the set
- Tracks 0:99; Head at track 75, requests for 23, 87, 36, 93, 66
- FCFS: 52 + 64 + 51 + 57 + 27 = 251 steps

Optimizing Seek Time (cont)

- Requests = 23, 87, 36, 93, 66
- SSTF: (75), 66, 87, 93, 36, 23
  - 9 + 21 + 6 + 57 + 13 = 106 steps
- Scan: (75), 87, 93, 99, 66, 36, 23
  - 12 + 6 + 6 + 33 + 30 + 13 = 100 steps
- Look: (75), 87, 93, 66, 36, 23
  - 12 + 6 + 27 + 30 + 13 = 87 steps

Optimizing Seek Time (cont)

- Requests = 23, 87, 36, 93, 66
- Circular Scan: (75), 87, 93, 99, 23, 36, 66
  - 12 + 6 + home + 23 + 13 + 30 = 90 + home
- Circular Look: (75), 87, 93, 23, 36, 66
  - 12 + 6 + home + 23 + 13 + 30 = 84 + home

Data Networks

- Technology focus includes protocols and software (more on this later ... Chapter 15 and beyond ...)

MS Disk Description

0x00: 0x02: <a jump instruction to 0x1e>
0x03: 0x0a: Computer manufacturer name
0x0b: 0x0c: Sectors per cluster (MS-DOS reads/writes a cluster of sectors)
0x0d: 0x0f: Reserved sector for the boot record
0x10: 0x10: Number of FATs
0x11: 0x12: Number of root directory entries
0x13: 0x14: Number of logical sectors
0x15: 0x15: Medium descriptor byte (used only on old versions of MS-DOS)
0x16: 0x17: Sectors per FAT
0x18: 0x19: Sectors per track
0x1a: 0x1b: Number of surfaces (heads)
0x1c: 0x1d: Number of hidden sectors
0x1e: ... Bootstrap program