Input/Output
Ch. 5.1–5.4, except 5.4.5

Block and Character Devices

Block devices have randomly-accessed blocks of data.

Character devices produce a sequential stream.

Not everything fits the categories well.
Bit-mapped displays

Huge Range of Speeds

Keyboard: A few bytes per second.

56K Modem: 7 KB/sec

IDE Disk: 5 MB/sec

Ethernet: 12.5 MB/sec

SCSI 2 Disk: 80 MB/sec

Gigabit Ethernet: 125 MB/sec

PCI bus: 528 MB/sec

Devices and Controllers

Separate the mechanical parts from the electronics.

Electronics are the controller or adapter.

CPU $\longleftrightarrow$ Adapter

Bytes, Op Codes, Interrupts, DMA

Adapter $\longleftrightarrow$ Device

Bits, Synchronization, Checksums

Communication may be constant.

May use analog signals.

Devices and Controllers

A controller may be able to handle several identical devices.

Controller-Device interface may be standard.
SCSI   *IDE*

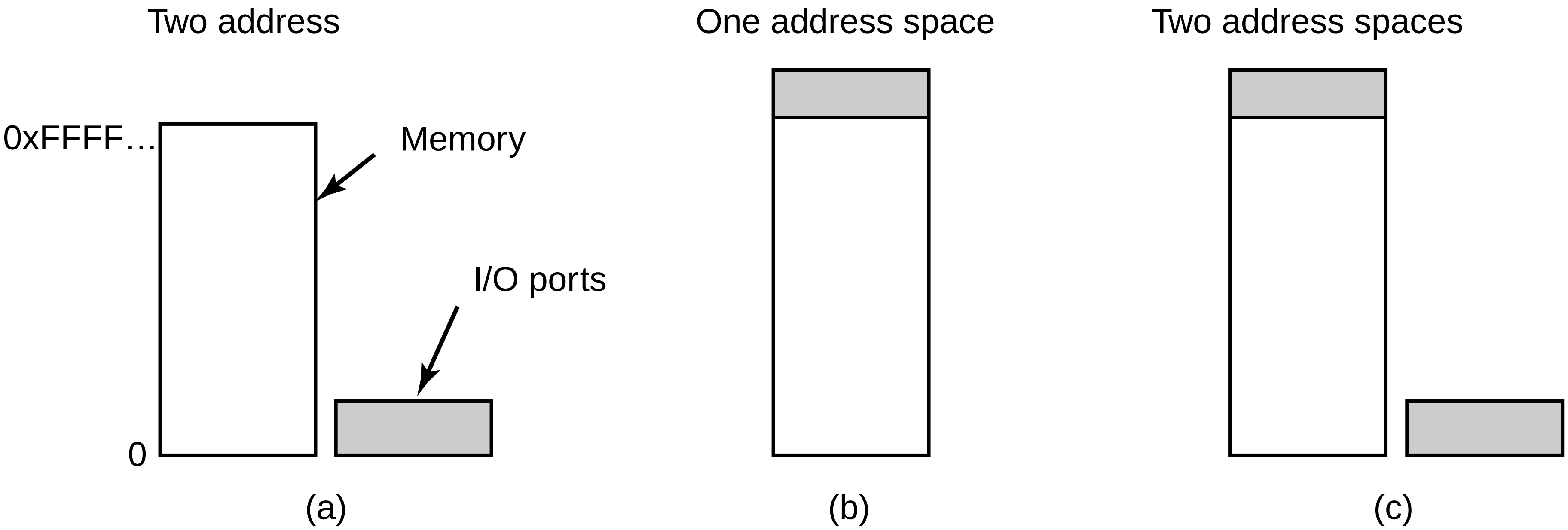
Talking to Devices

Devices have special registers which control them.

The CPU reads or writes these registers.

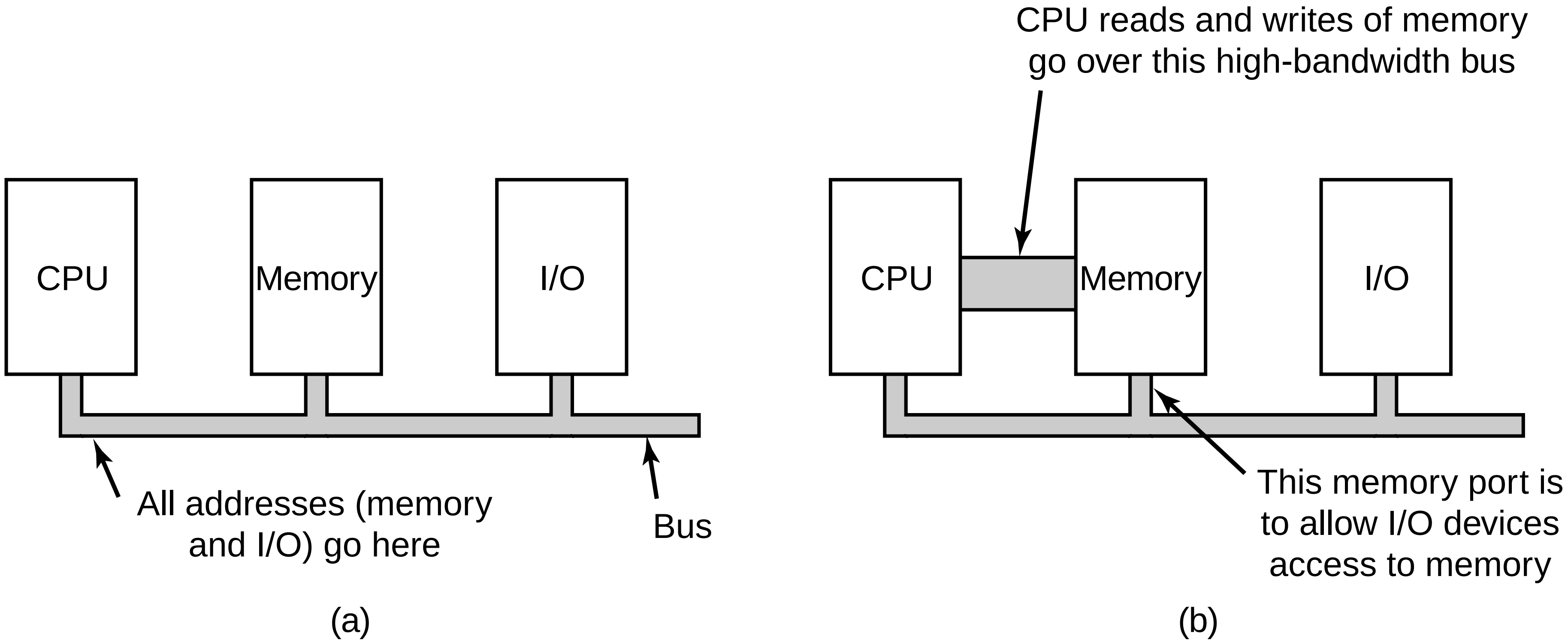
Send and receive data.
Query status.
Send addressing information
Set operating modes.

Addressing Devices



IN and OUT LOAD and STORE Either

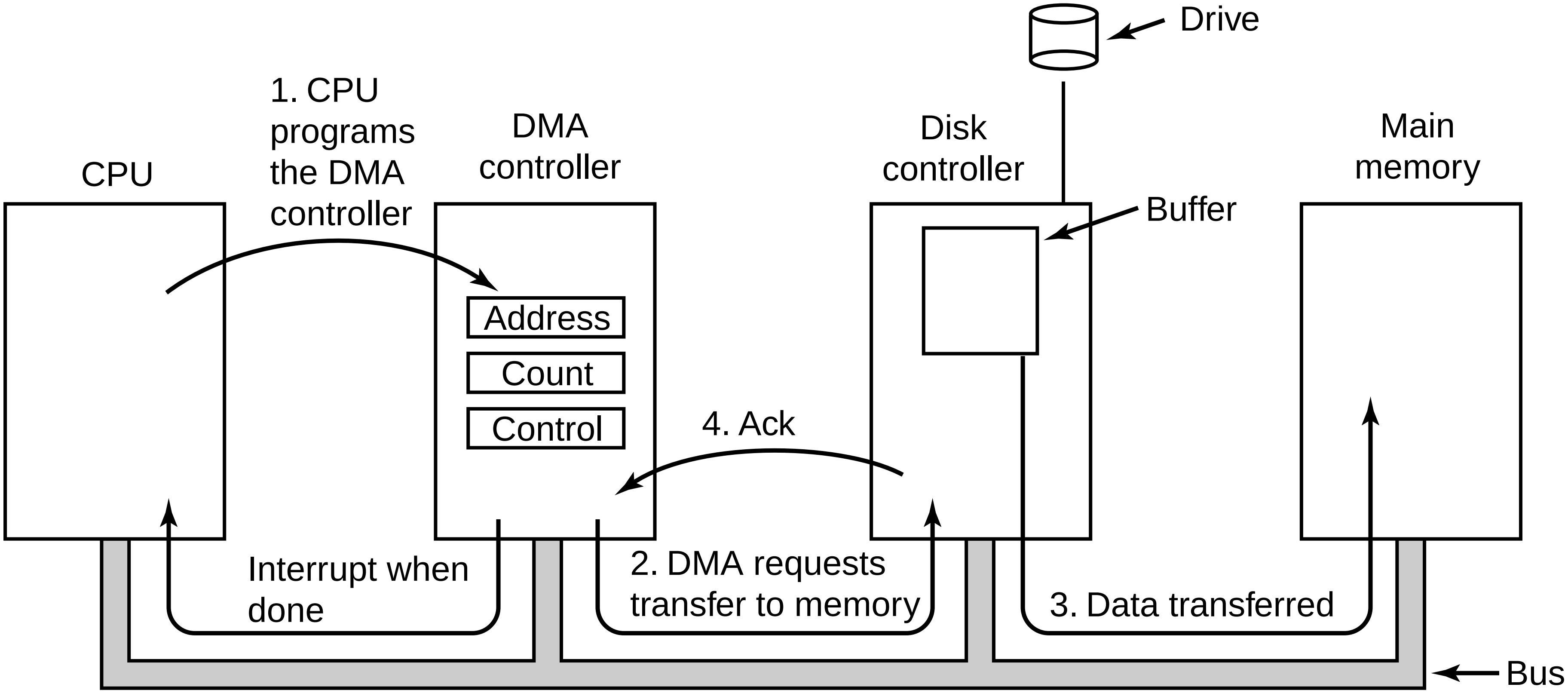
Multiple Buses


CPU reads and writes of memory go over this high-bandwidth bus
CPU
Memory
I/O
CPU
Memory
I/O
All addresses (memory and I/O) go here
Bus
This memory port is to allow I/O devices access to memory
(a)
(b)

Buses are optimized for their application.

PC's typically have three.
memory, PCI, ISA

Direct Memory Access



I/O Types

Direct

Interrupt-Driven

DMA

DMA

Data sent between I/O device and memory.

Data sent to DMA then to memory (or wherever).
Allows transfers between devices.

Interrupts

When an interrupt occurs, CPU transfers
to a standard location.

Locations generally listed in an *interrupt vector.*
Controlled by CPU.

Each interrupt delivers a type code which indexes the vector.

Registers are saved on a stack.
Usually the kernel's stack.

Pipelines, esp. multiple, complicate figuring
out where you stopped.

I/O System Goals

Device Independence

Program for I/O, I'll tell you what device later.

Uniform Naming

Devices names do not depend on the device.

Error Handling

Don't pass errors up to caller if can be avoided.

Synchronous Asynchronous

Whether I/O requests block until completion.

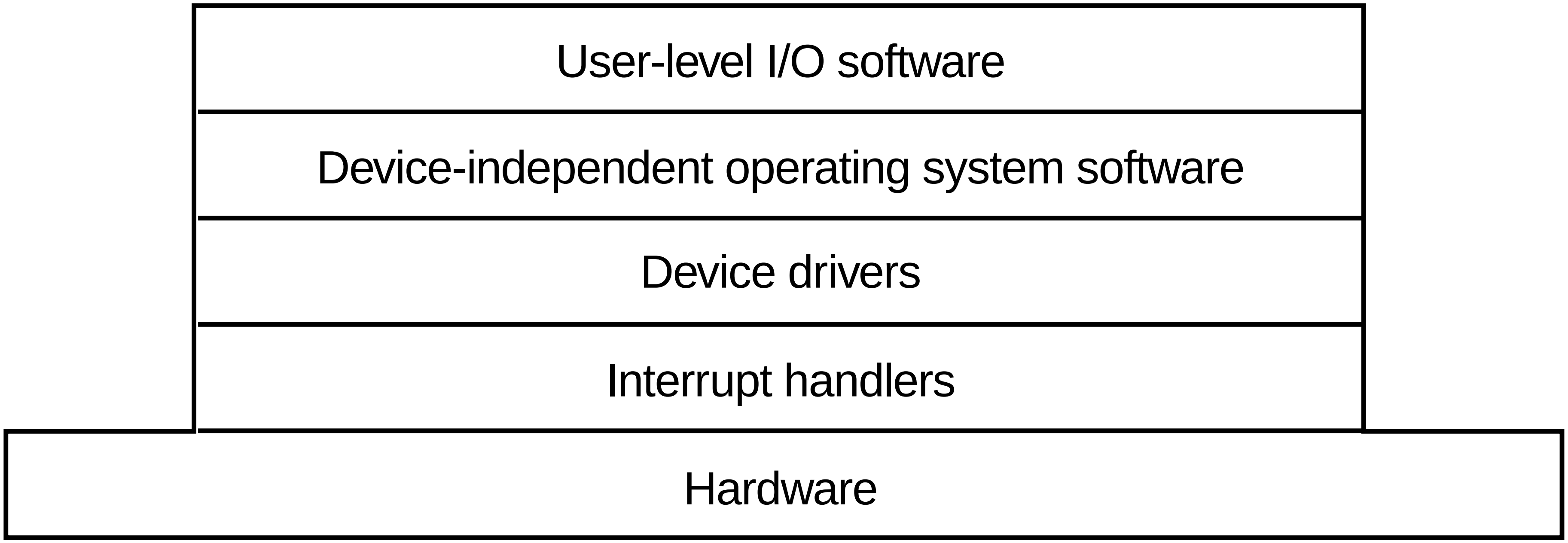
Frequently both are provided.

Buffering

Collect data generated before delivery.

Layers

Like any complex software, the I/O system is built in layers.



User-level I/O software

Device-independent operating system software

Device drivers

Interrupt handlers

Hardware

Interrupts

Device drivers do something that will cause an interrupt, then sleep.

The interrupt handler does the minimum, then awakens the sleeping driver.

Save regs, *Mem context for service func,*
Stack for service func, *Ack interrupt,*
Copy regs to proc descr., *Run service func,*
Choose process, *Set context for process,*
Load regs, *Run new process*

Drivers

Takes enqueued abstract read and write requests and performs them against particular hardware.

Knows the details of the specific device.

Writes needed commands to the hardware registers.

If it needs to wait, it can suspend itself until the interrupt.

May have several requests outstanding;
Will have to figure out which one that interrupt is for.

Drivers, Cont.

Each device, or group of very similar devices, needs its own driver.

Usually have a standard interface to the rest of the O/S.

Useful to add new drivers.

Interface used only by O/S programmers.
But maybe by other ones than wrote the OS.

Device-Independent Parts

Some driver software is device-independent.

Uniform interfacing for device drivers
Buffering
Error reporting
Allocating and releasing dedicated devices
Providing a device-independent block size

Getting Drivers Into The OS

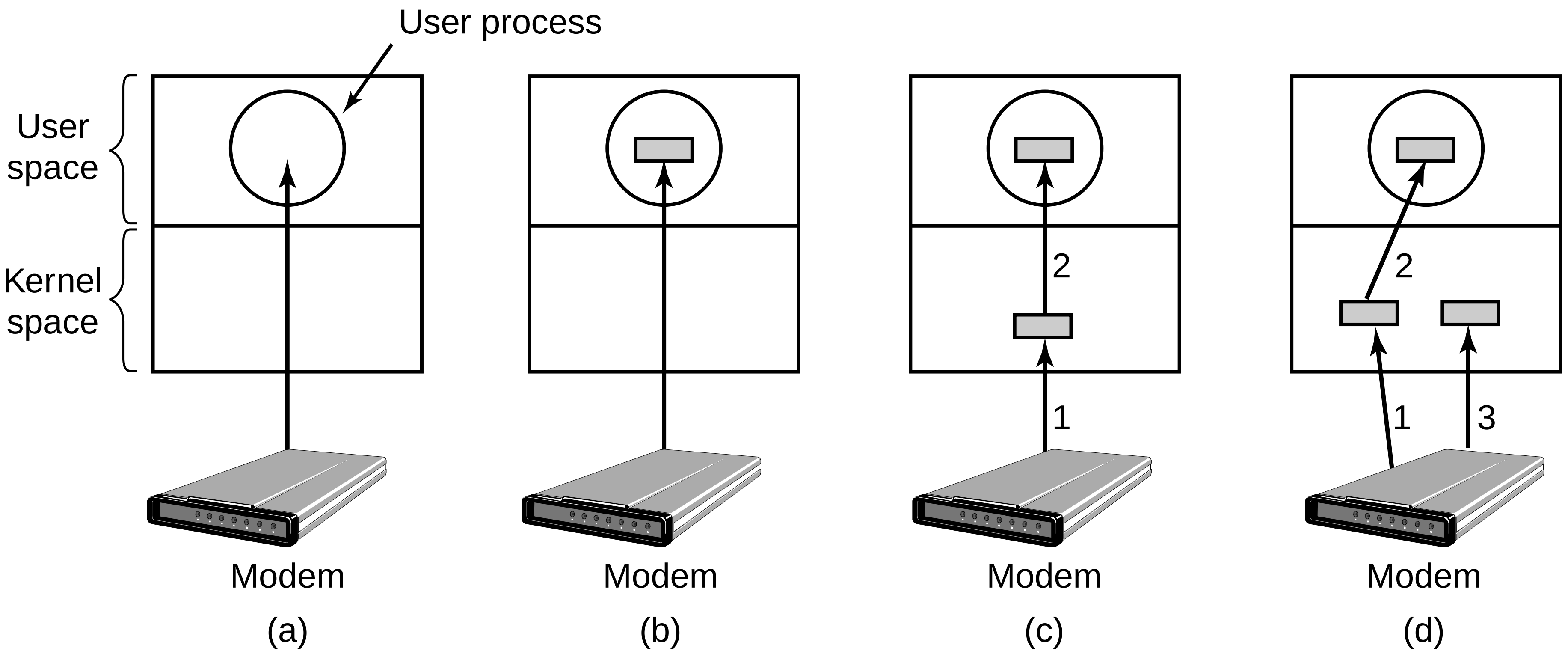
Traditional: Just compile or link the O/S.

Need to rebuild the O/S whenever you get new hardware.

Not very practical with PC's
Often don't have the source
Think of Aunt Maude compiling a kernel

Present drivers can be added dynamically to the O/S.

Buffering



For disks, buffer contents may be retained for reuse, or use by other processes.

Errors

Impossible request: Just report.

Device failure: Report, possibly halt O/S.

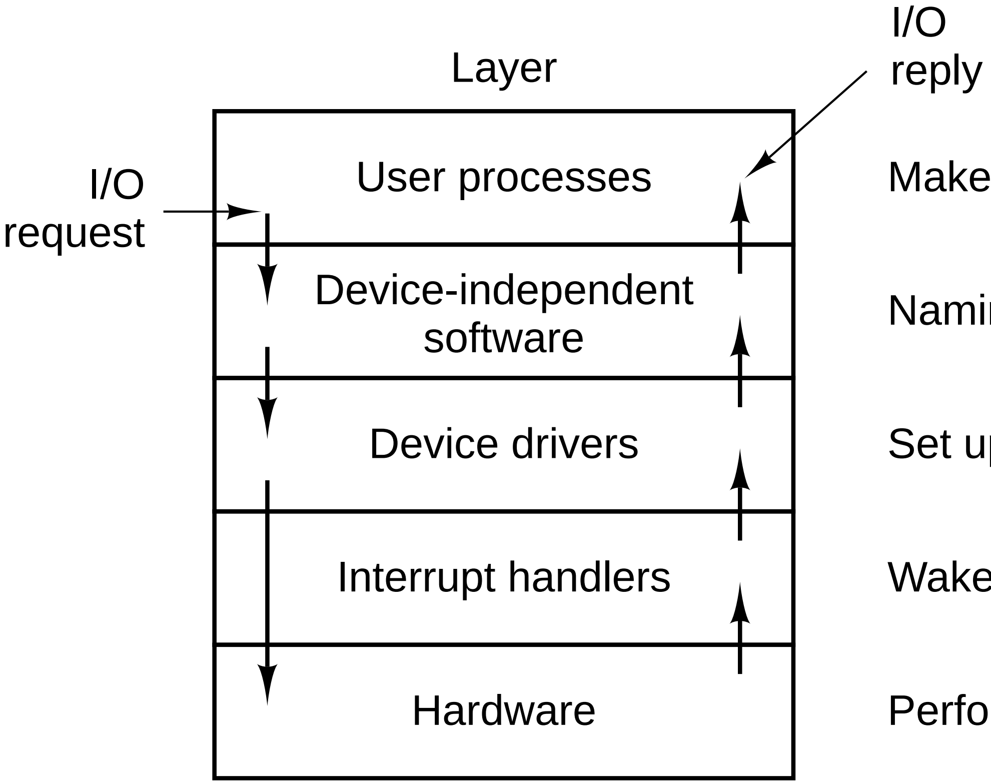
User Space

Some I/O processing is done in userland.
Usually libraries or programming languages.

Formatting, base conversion.

Spooling.

I/O Layers, Again

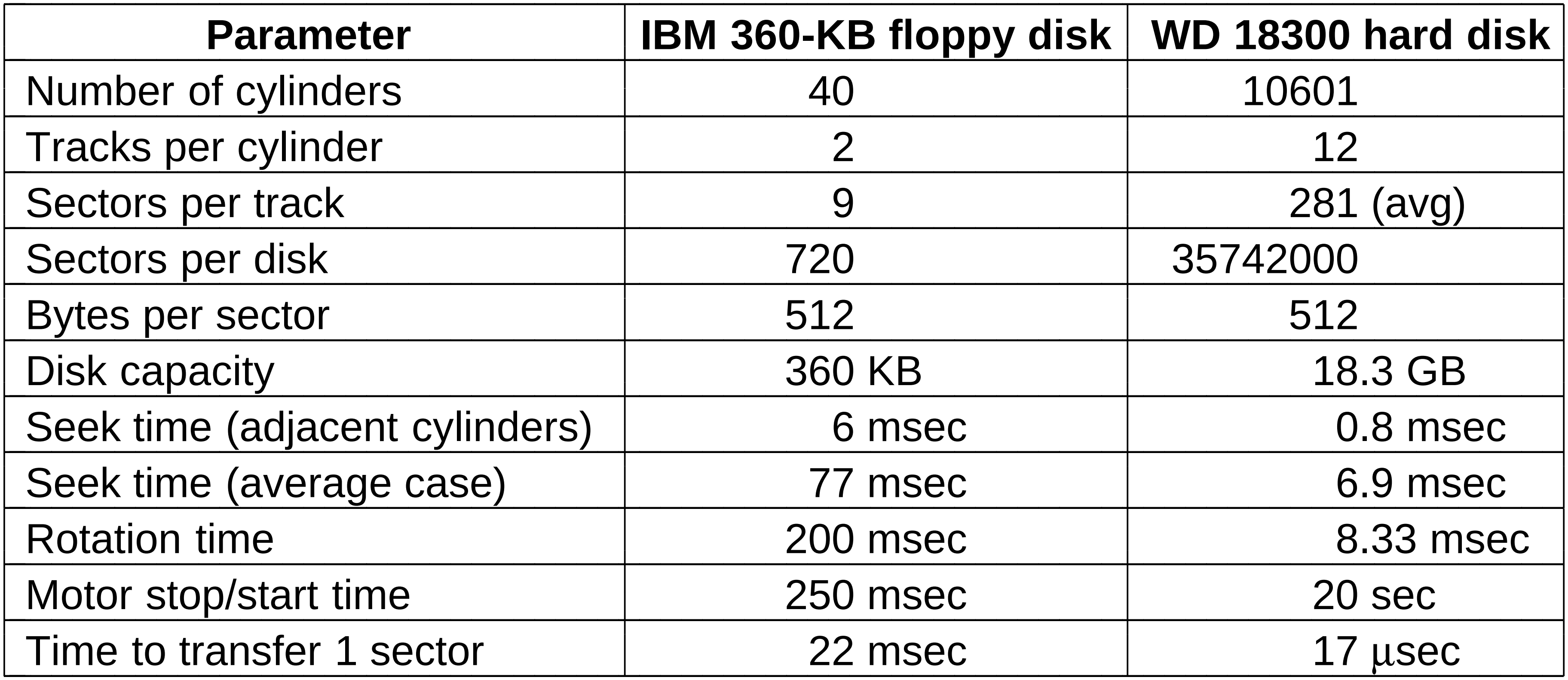


I/O request

I/O reply

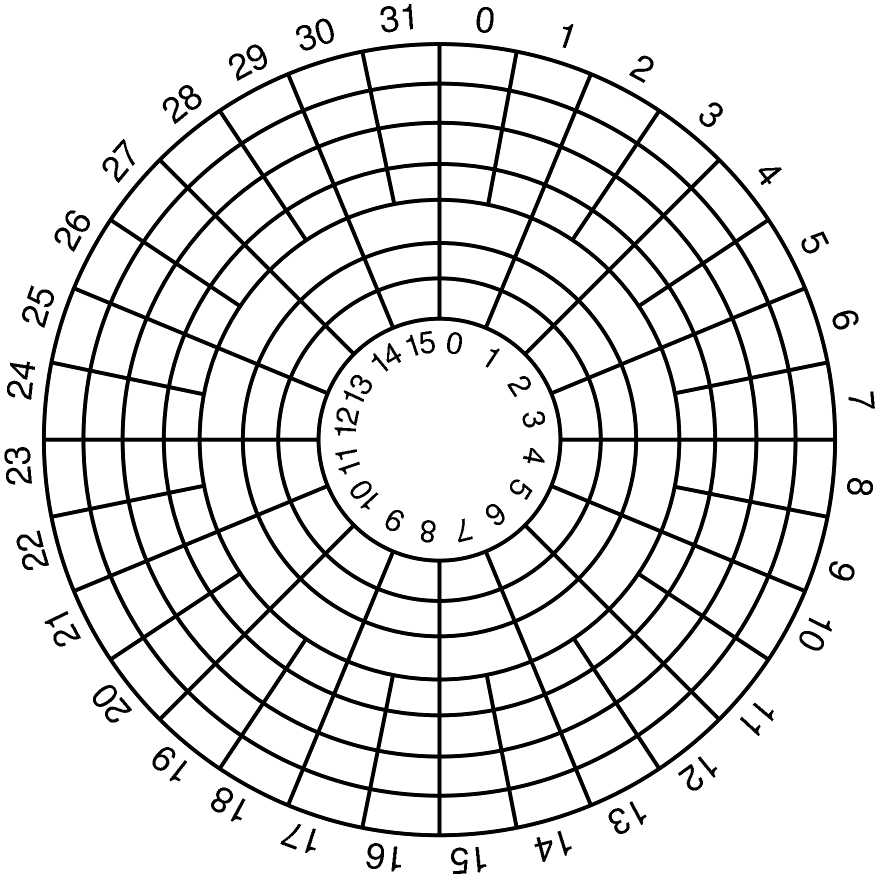
Layer	I/O functions
User processes	Make I/O call; format I/O; spooling
Device-independent software	Naming, protection, blocking, buffering, allocation
Device drivers	Set up device registers; check status
Interrupt handlers	Wake up driver when I/O completed
Hardware	Perform I/O operation

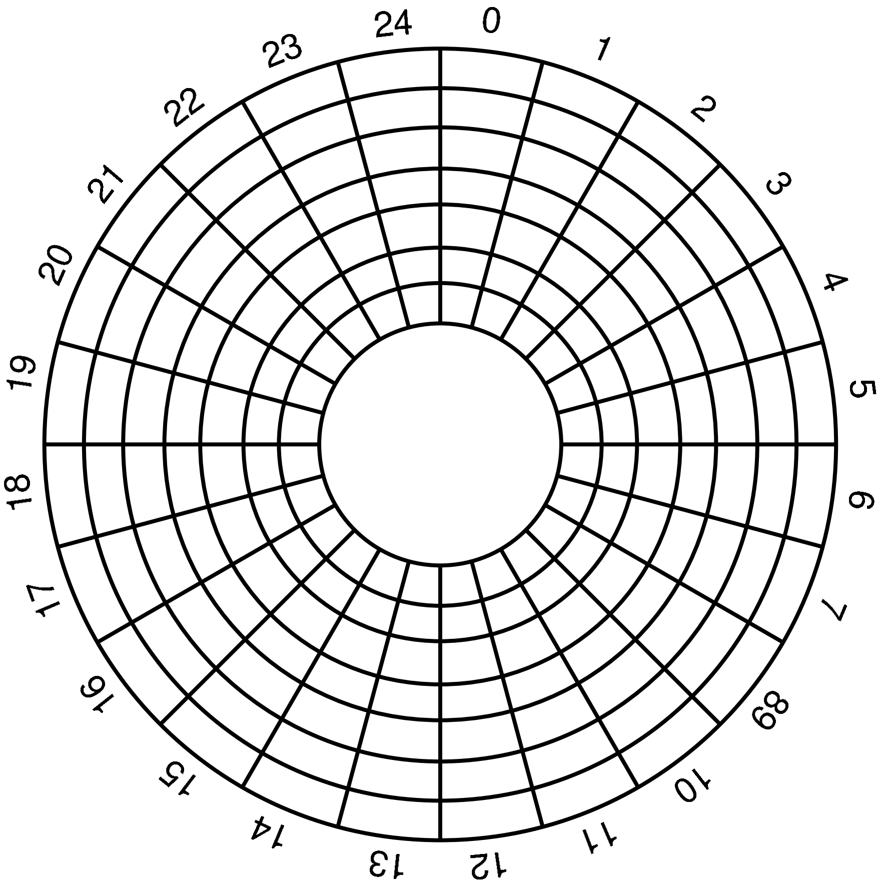
Disks



Parameter	IBM 360-KB floppy disk	WD 18300 hard disk
Number of cylinders	40	10601
Tracks per cylinder	2	12
Sectors per track	9	281 (avg)
Sectors per disk	720	35742000
Bytes per sector	512	512
Disk capacity	360 KB	18.3 GB
Seek time (adjacent cylinders)	6 msec	0.8 msec
Seek time (average case)	77 msec	6.9 msec
Rotation time	200 msec	8.33 msec
Motor stop/start time	250 msec	20 sec
Time to transfer 1 sector	22 msec	17 μsec

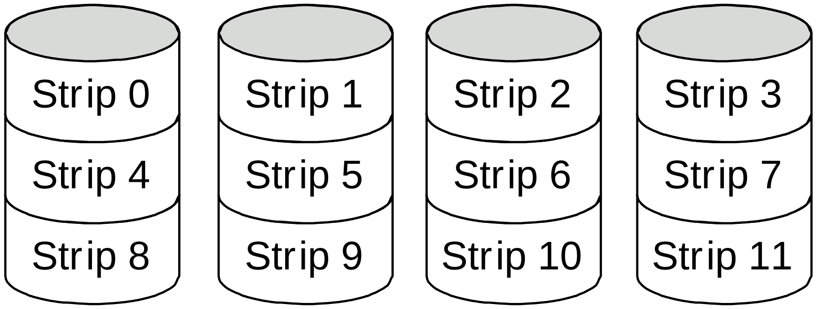
Sectors


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15


0
1
2
3
4
5
6
7
89
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

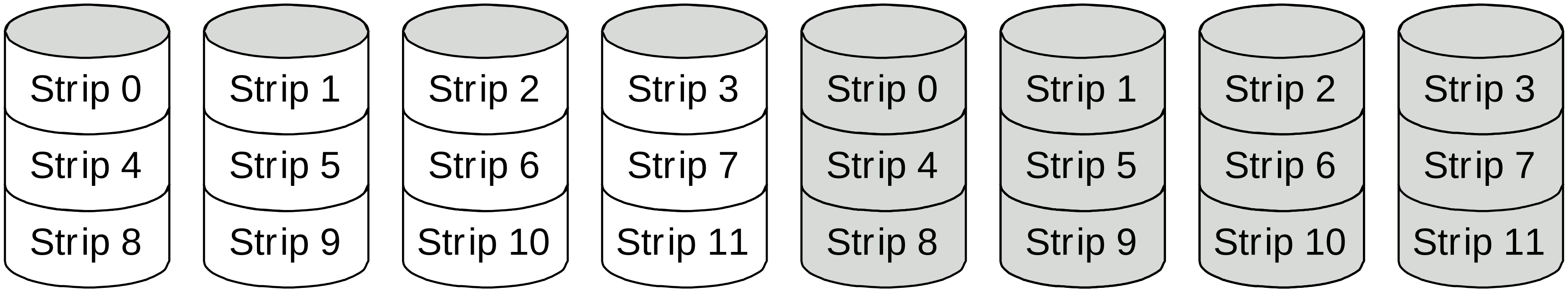
RAID

0: Striping



Strip 0	Strip 1	Strip 2	Strip 3
Strip 4	Strip 5	Strip 6	Strip 7
Strip 8	Strip 9	Strip 10	Strip 11

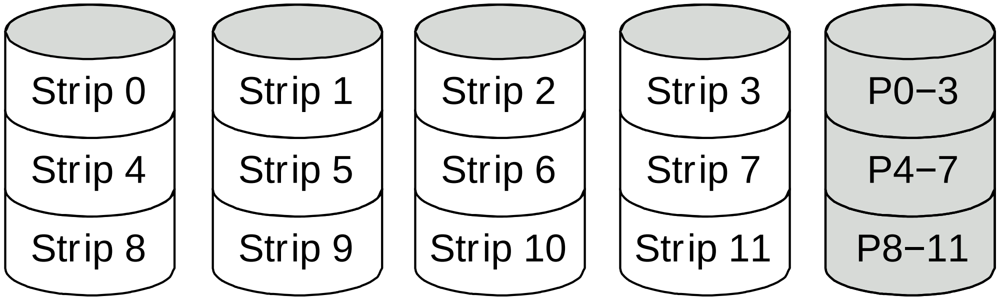
1: Copying



Strip 0	Strip 1	Strip 2	Strip 3	Strip 0	Strip 1	Strip 2	Strip 3
Strip 4	Strip 5	Strip 6	Strip 7	Strip 4	Strip 5	Strip 6	Strip 7
Strip 8	Strip 9	Strip 10	Strip 11	Strip 8	Strip 9	Strip 10	Strip 11

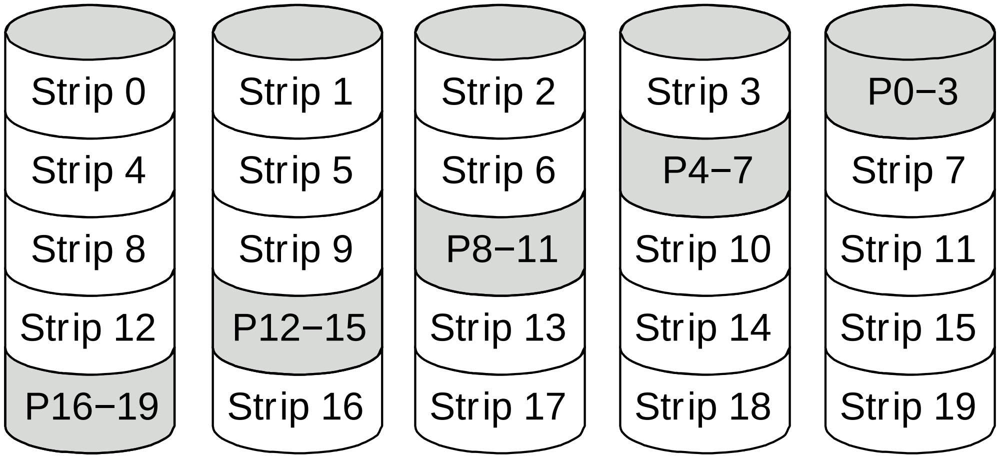
RAID

4: Striping with parity



Strip 0	Strip 1	Strip 2	Strip 3	P0−3
Strip 4	Strip 5	Strip 6	Strip 7	P4−7
Strip 8	Strip 9	Strip 10	Strip 11	P8−11

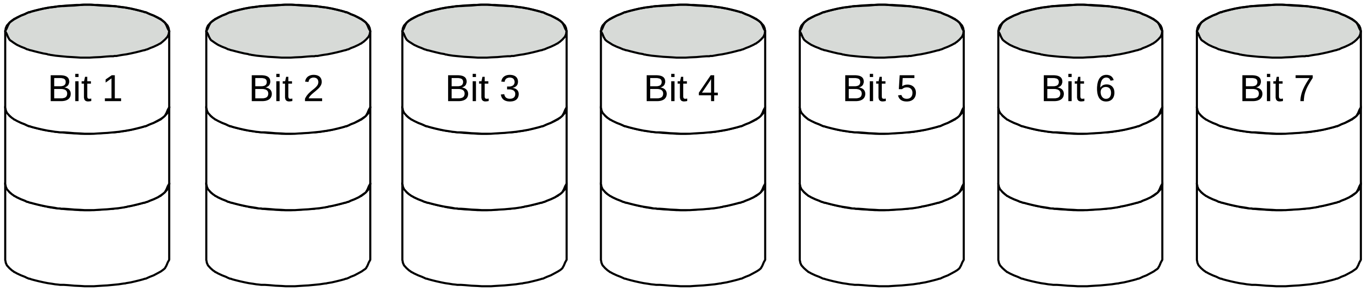
5: Distributed parity



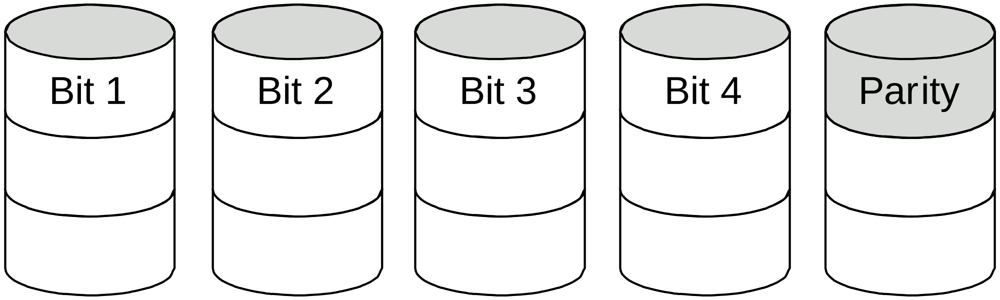
Strip 0	Strip 1	Strip 2	Strip 3	P0−3
Strip 4	Strip 5	Strip 6	P4−7	Strip 7
Strip 8	Strip 9	P8−11	Strip 10	Strip 11
Strip 12	P12−15	Strip 13	Strip 14	Strip 15
P16−19	Strip 16	Strip 17	Strip 18	Strip 19

RAID

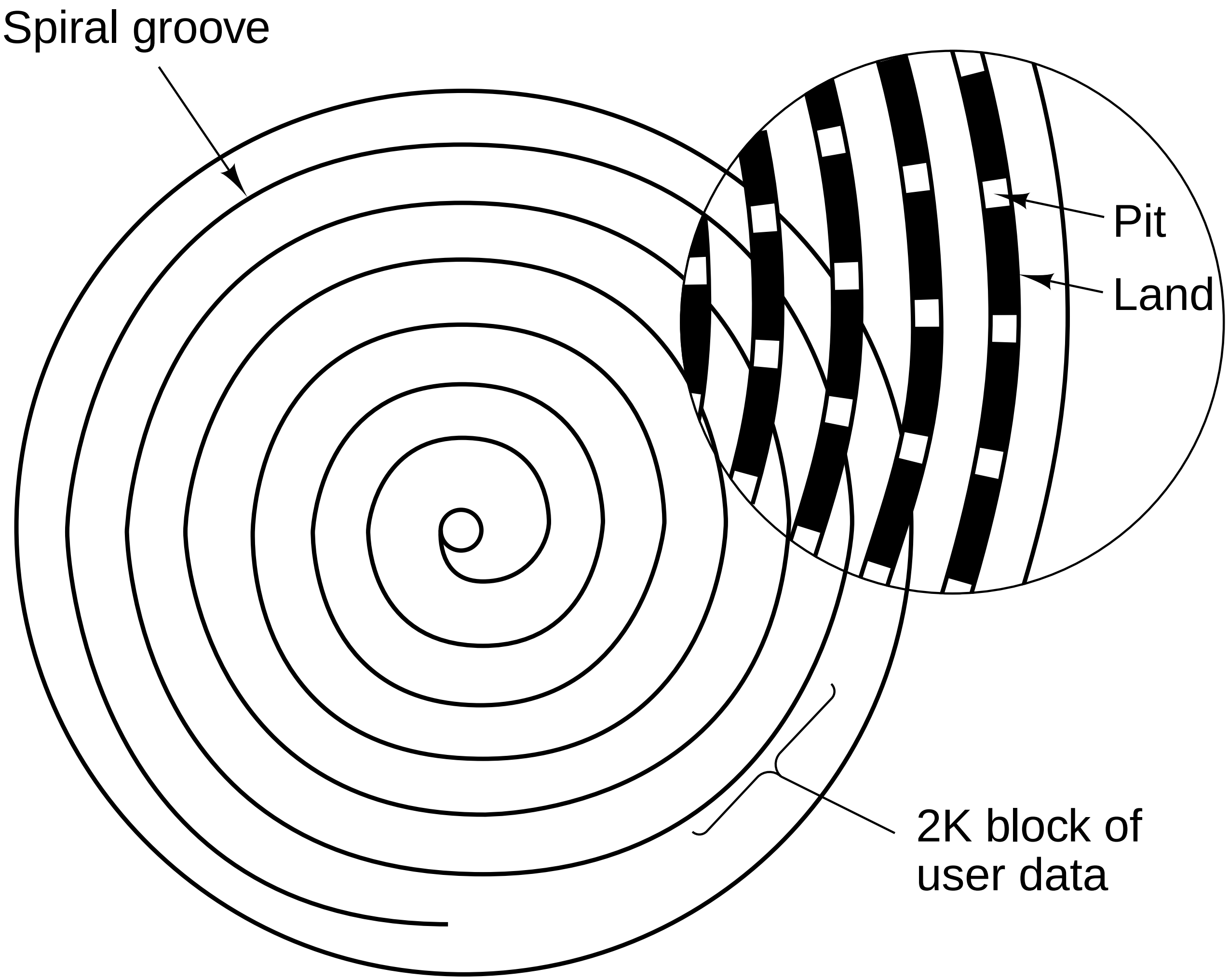
2: Bit splits with Hamming code



3: Bit splits with parity



CD ROMS



CD ROM Mechanics

Compact Disks designed for recording music.
1980 - Red Book
Philips/Sony

Bits are indicated with pits in the surface.
Lack of a pit is a *land.*
Read by a laser.

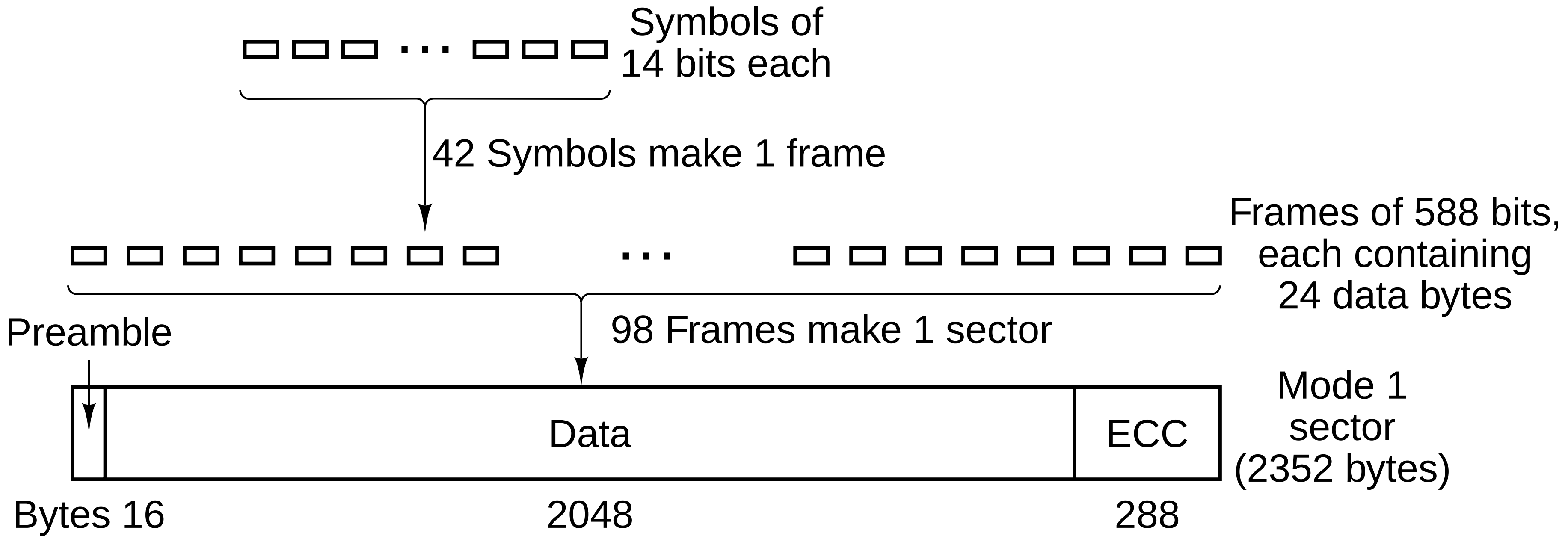
Disk rotation rate must vary with head location to produce a fixed data rate.
Constant linear velocity

Hard disks use constant angular velocity, 3600 or 7200 rpm.

CD-ROMs vary from 530 rpm to 200 rpm.

CD ROM Formats

1984 - Yellow Book



Each byte is coded in 14 bits with extensive error correction in hardware.

CD ROMs

Mode 2: For audio and video, use ECC space for data.

Single speed: 75 sectors/sec = 153,600 bytes/sec

1986 - Green Book. Audio, video, data in same sector.

File System

ISO 9660

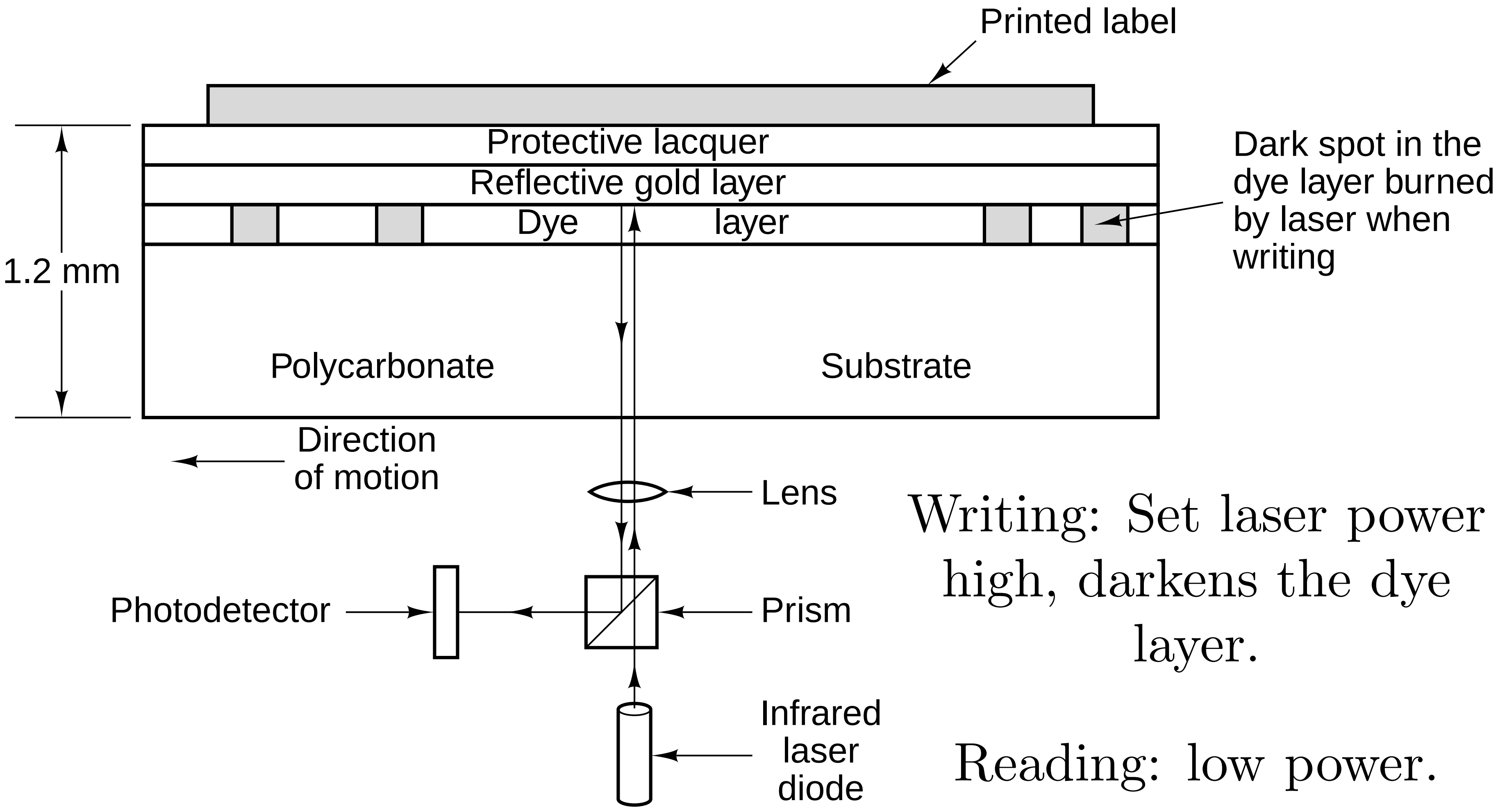
Level 1: DOS Names
Directories to 8 levels
Contiguous files

Level 2: 32 characters.

Level 3: Not contiguous.

Rock Ridge: Unix extensions.

Recordable CDs



Writing: Set laser power high, darkens the dye layer.

Reading: low power.

Recording CDs

1989 - Orange Book
CD-ROM XA: *Incremental CD Writing*

Before: Single VTOC
Orange: Each track has a VTOC
Use the most recent VTOC

Appear to delete by leaving things off a new track VTOC

Multisession CDs.
Each track must be written at one time.

CD-Rewritables

Dye layer can be changed between two states.

High power marks the dye.

Medium power erases it.

Low power reads it.

DVDs

Similar to CD's.
Higher-frequency laser allows closer packing.

Smaller pits: 0.4 microns v. 0.8

Tighter spiral: 0.74 microns v. 1.6

Red laser 0.65 microns v. 0.78

Double-layer format: stack two single layers.
Laser focused differently for each layer.

Double-sided formats also.
Turn 'em over.

Hard Disk Formatting

Low-level formatting.
Mark out and number the sectors.

Partitioning
Divide into virtual disks via a partition table.

High-level formatting.
Create an empty filesystem.

Low-Level Formatting

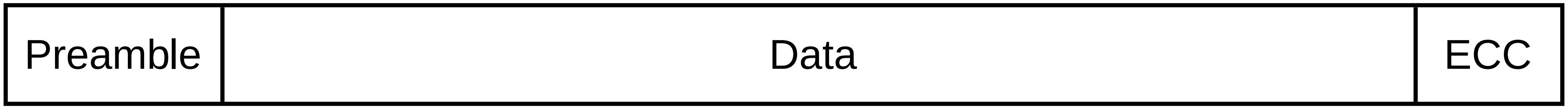
New disk is a stack of round plates coated
with magnetic material.

Low-level formatting fills each track with sectors.
Small separation between.

Usually done by manufacturer.

Low-level format may consume 20% of hardware capacity.

Sector Format



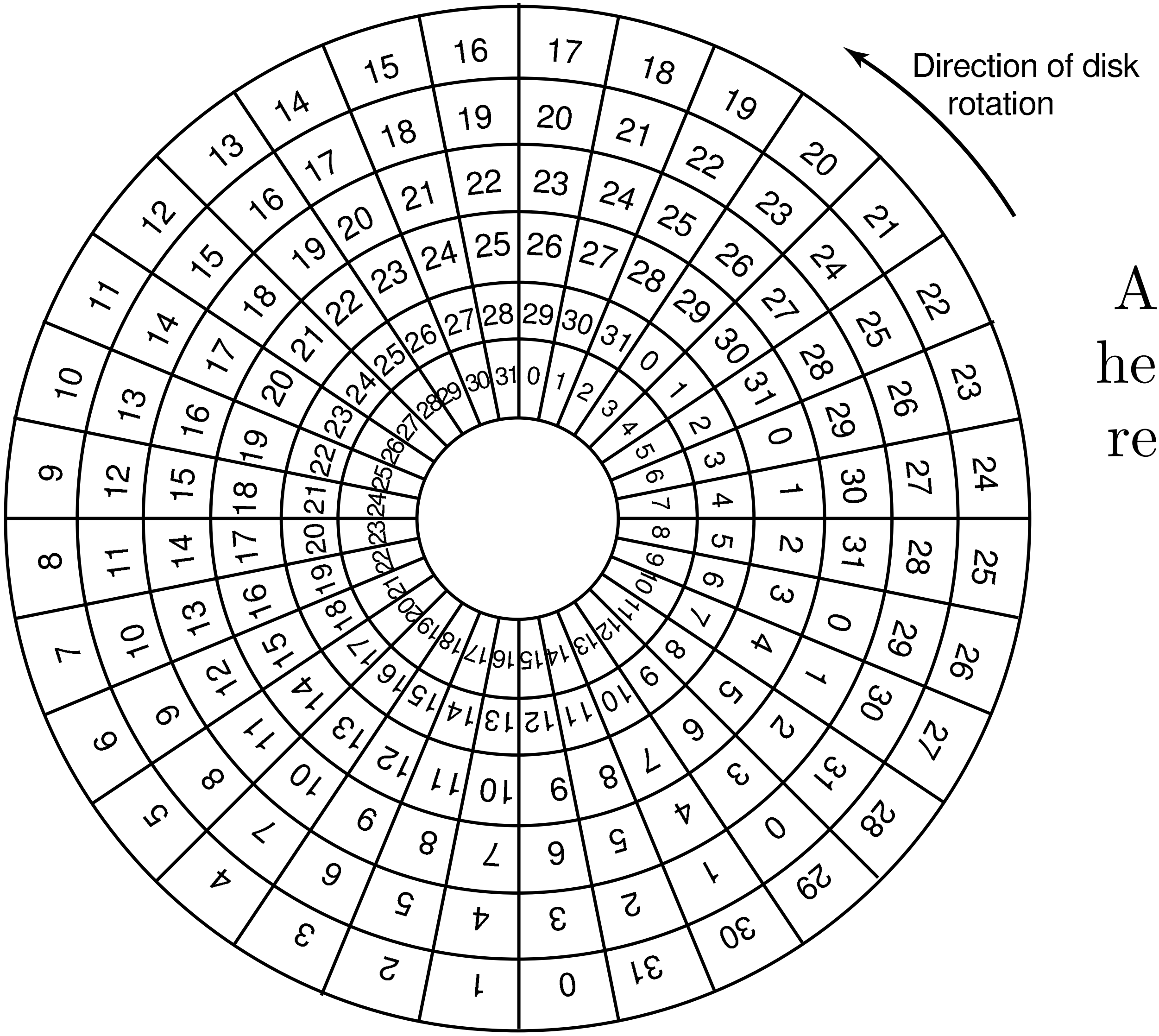
Preamble

Data

ECC

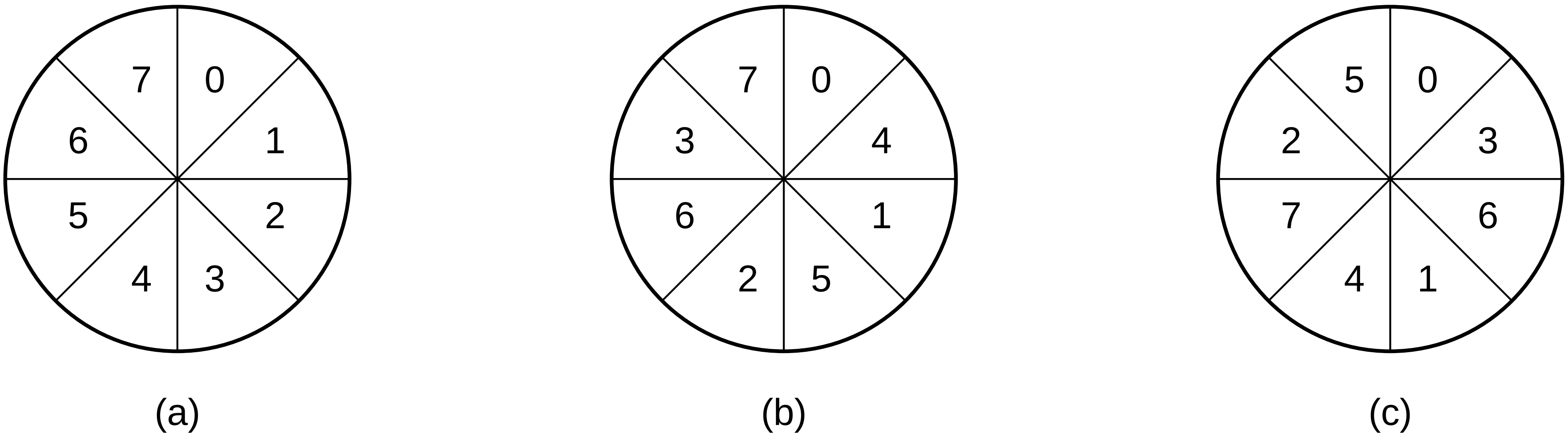
Preamble contains a recognizable pattern, along with a sector number and control info.

Cylinder Skew



Allow a delay for head motion when reading sectors in order.

Interleaving



Allows newly-read sector to be copied out without having to go around again.

Partitioning

Reserve a sector as the the Master Boot Record (MBR)
Usually zero.

Loaded and run when the system boots.

Rest is divided into partitions.
Virtual disks.

Partition table records this division.

PC partition tables have room for four partitions.
Also extended partitions partition a partition.

High-Level Format

Fill the boot block.

Free block list.

Root directory.

Disk Times

Seek: Find the track.

Rotational delay: Wait for the sector.

Transfer time.

Seek time generally dominates.

Disks requests need not be filled in order.
Usually optimized to reduce seek time.

Disk Scheduling Algorithms

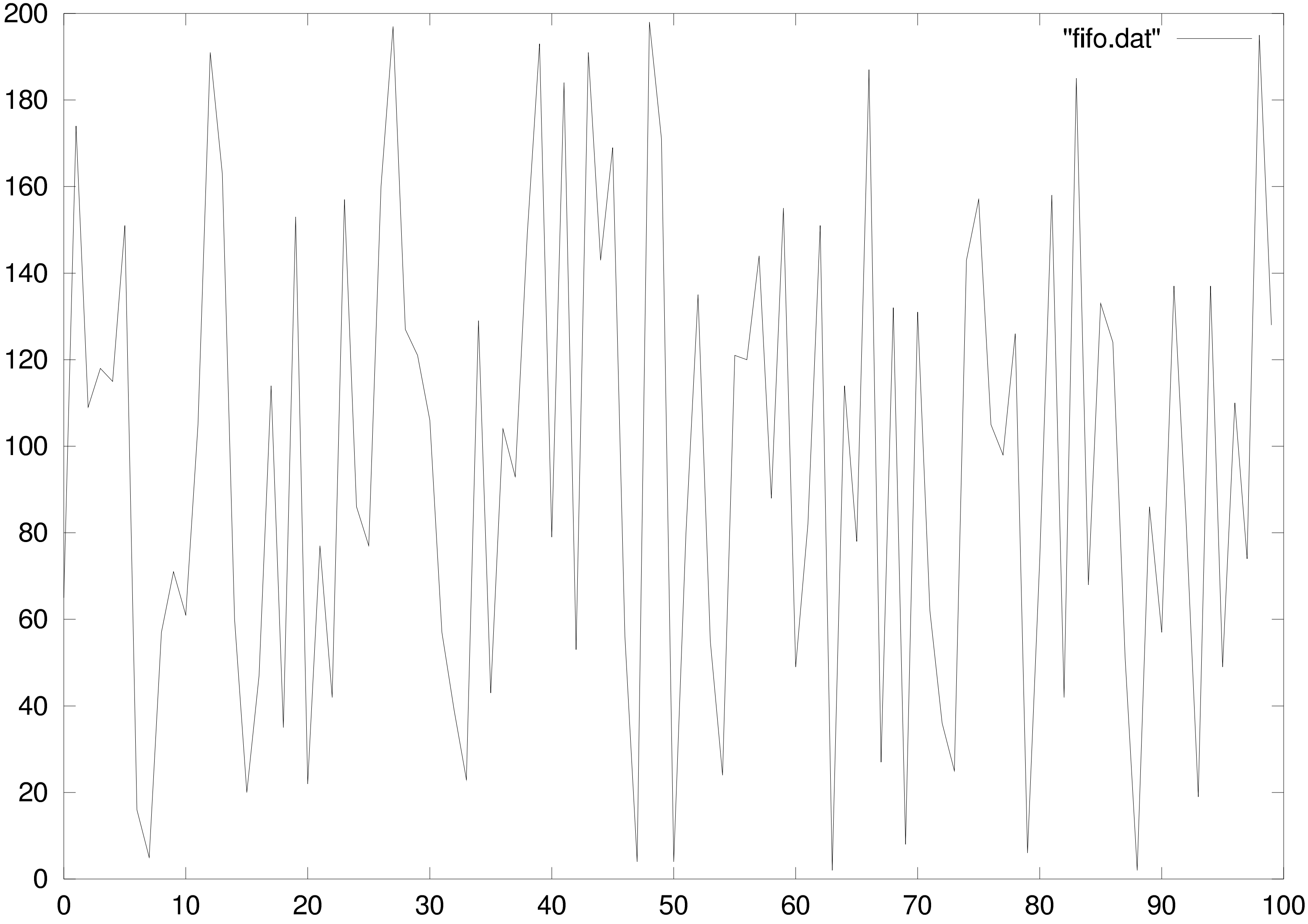
First-Come First Served (FCFS).

Shortest Seek-Time First (SSTF).
Always move to nearest request.

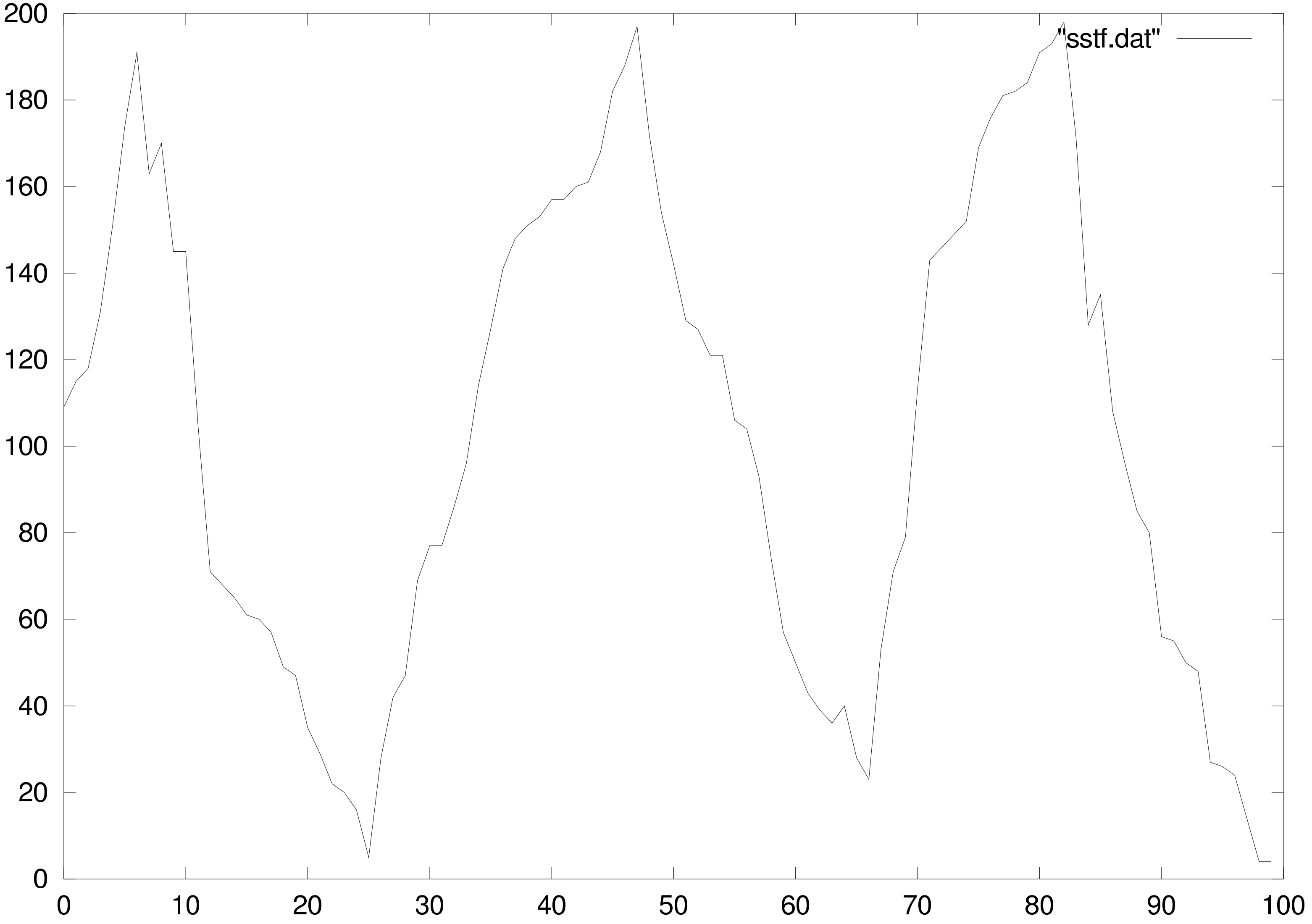
Elevator.
Keep going in the current direction.
Also called SCAN.

One-Way Elevator
Process requests while going only one way.
Also C-SCAN.

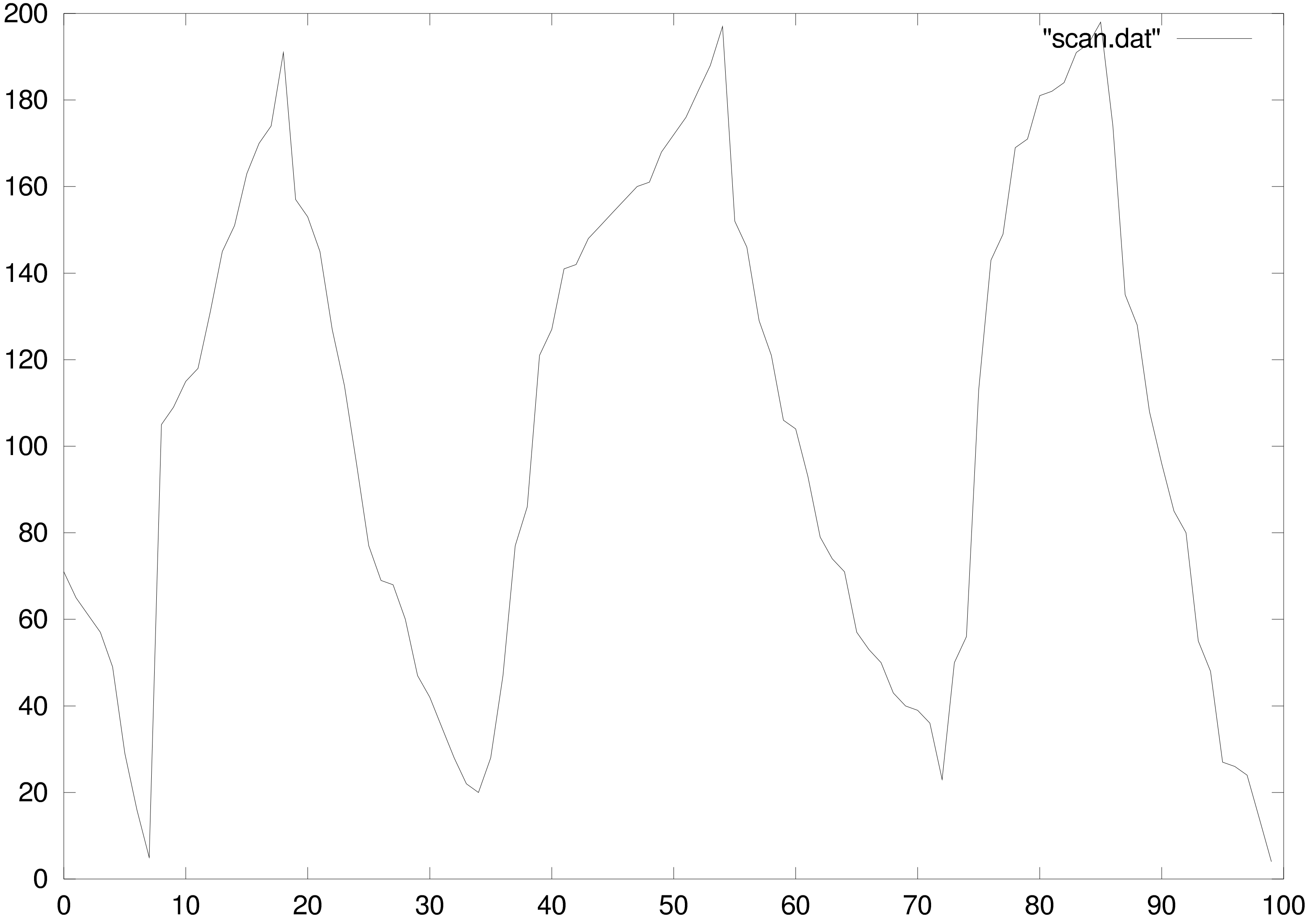
FCFS 100 Random Requests



SSTF 100 Random Requests



Elevator 100 Random Requests

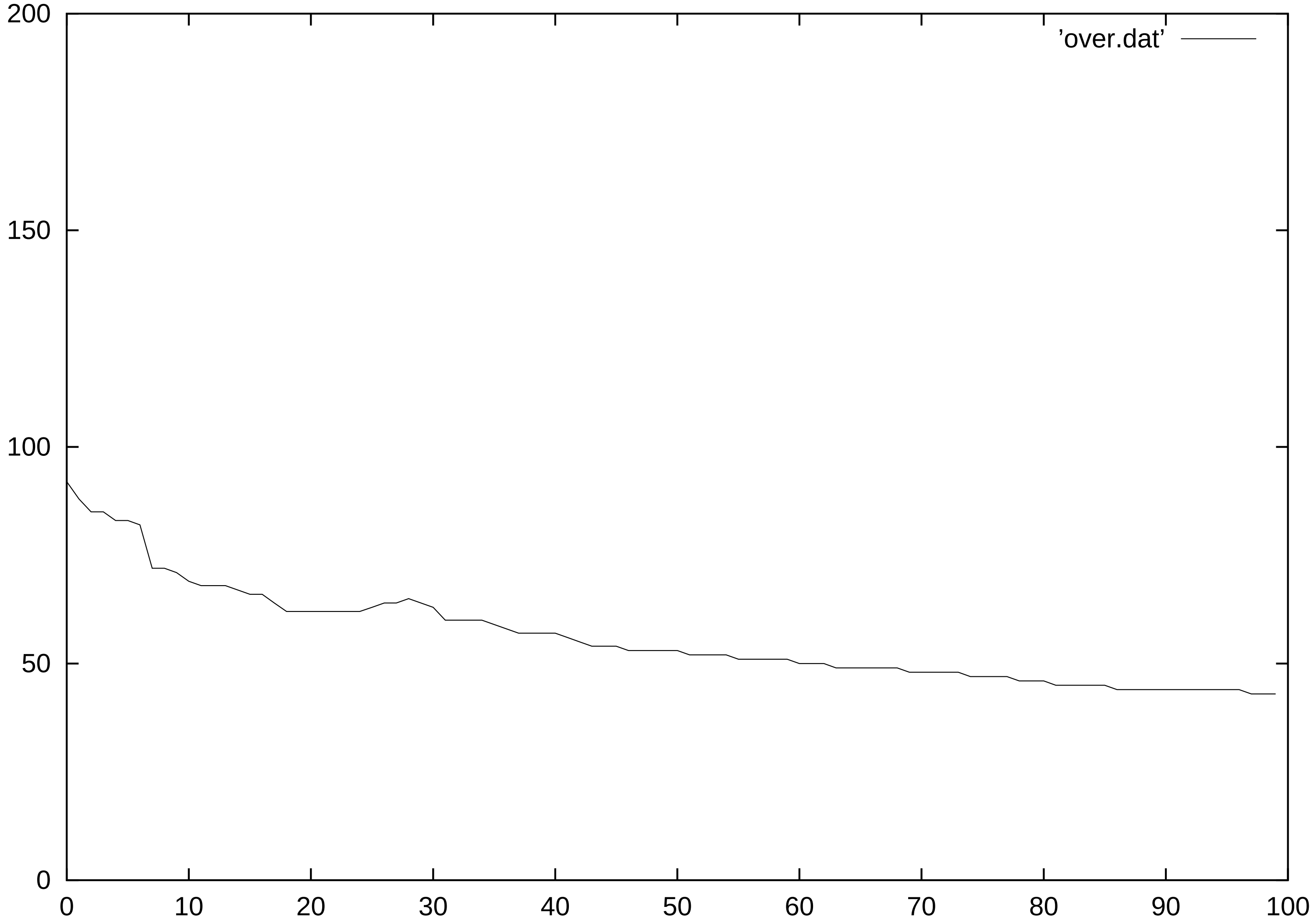


Costs

FCFS	6844
SSTF	1084
Elevator	1172

Why is Elevator preferred: Behavior under heavy loads.

SSTF Overloaded (12 arrivals/service)



Elevator Is Unfair

A random request in the middle of the disk must wait at most the scan width for service.
Average half the width.

A random request in the edge of the disk must wait at most twice the scan width for service.
Average one width.

The one-way elevator eliminates this discrepancy.

Disk Errors

Disks are manufactured with increasing data density.

Impossible to manufacture a flawless disk.
Some sectors will not correctly return data stored on them.

Bad sectors are detectable by failure of the ECC check after read.

Low-level formatting omits non-functional sectors.
Reserves spares for later failures.

Disk Errors

When a sector read fails in operation it is reread.

If it fails too many times, the controller will replace it with a spare.

All invisible to the O/S.

OS may use similar techniques if the disk runs out of spares.

Traditionally, O/S kept track of bad sectors.
Job largely moved to the controller.

Seek Errors

If the head is not where it should be after a
seek, must be recalibrated.
Head is no longer where the controller thinks it is.

Recalibration moves the head all the way to the edge.
Now the controller knows where it is.

Hard drives usually done by the controller.
Floppies done by the O/S.

Sources

Tanenbaum, *Modern Operating Systems*
(Course textbook.)