Input/Output

Week 12 – Tutorial

Problem 5.16

- Why are output files for the printer normally spooled on disk before being printed?

Problem 5.16 – Solution

- Imagine, that files are printed immediately by a process. In such a case the following situation might occur:
  - The process captures printer and prints several symbols
  - Then for some reason, the process goes to sleep
  - Thus, printer is blocked until the process is running again and the other processes that might need to print would wait unnecessarily

Problem 5.22 (1/9)

- Compare RAID level 0 through 5 with respect to read performance, write performance, space overhead, and reliability

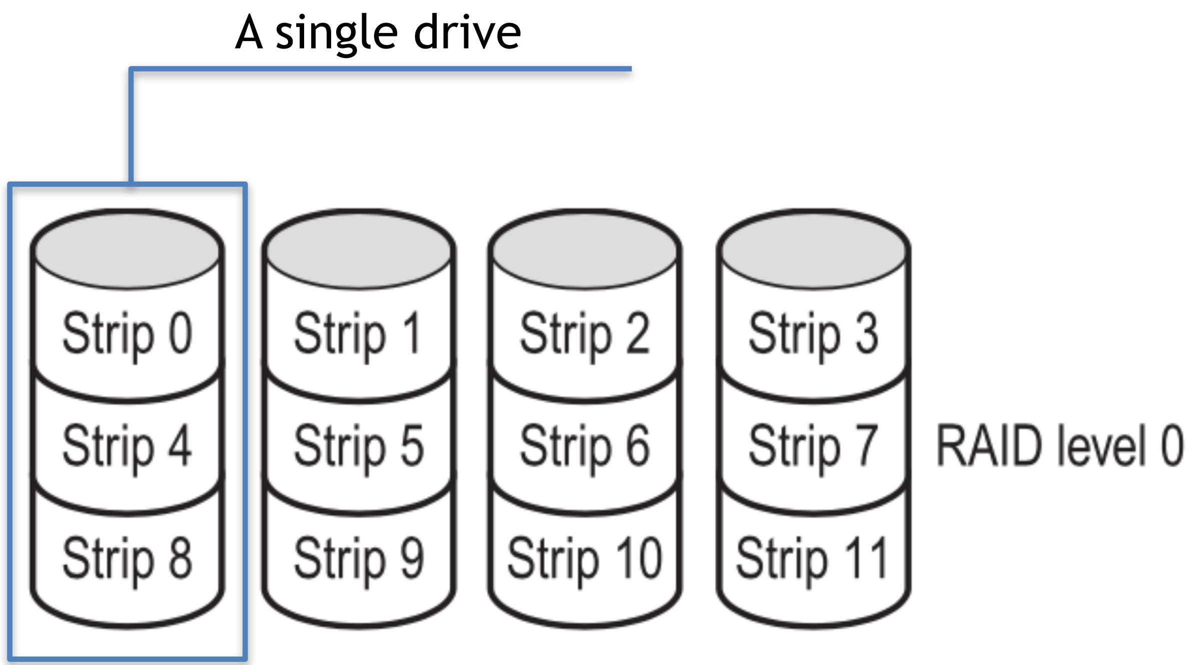
Problem 5.22 (2/9)

- RAID: Redundant Array of Inexpensive (or Independent) Disks
- Striping: distributing the data across several disks
- Parity: a technique that is used to detect data losses and allows data recovery in some cases

Problem 5.22 (3/9)

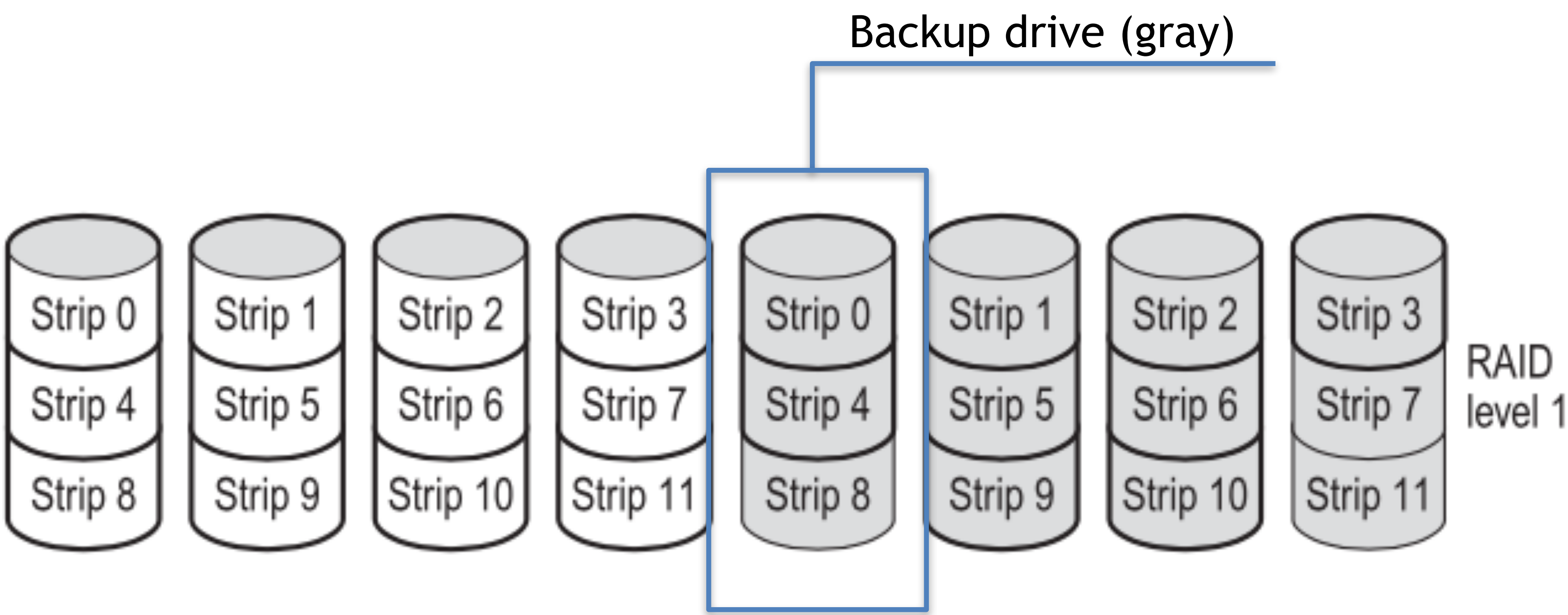
RAID	Description
0	Striping only, no mirroring, no parity checks
1	Mirroring with or without striping; no parity checks
2	Bit-level striping with Hamming-code based parity; one byte is split into two parts, three parity bits are added; resulting bits are written to seven disks, one bit per disk (all disks’ rotation is synchronized)
3	Bit-level striping with a parity bit computed for each word and stored on dedicated parity drive
4	Strip-for-strip parity: all the strips are XOR’ed together, resulting in a parity strip
5	The same as RAID 4, except that parity bits are distributed across all the drives in round-robin fashion

Problem 5.22 (4/9)



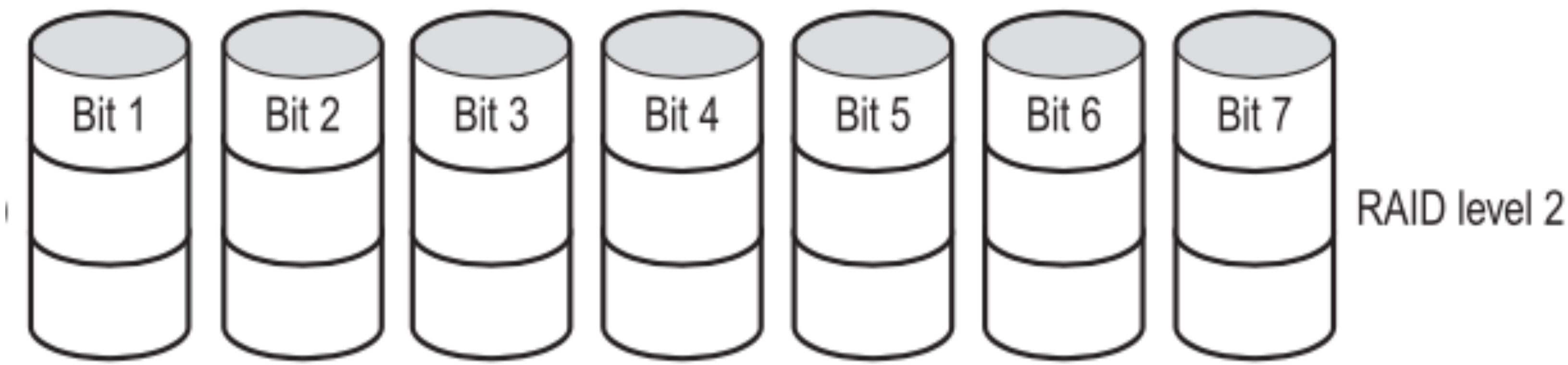
RAID 0

Problem 5.22 (5/9)



RAID 1

Problem 5.22 (6/9)

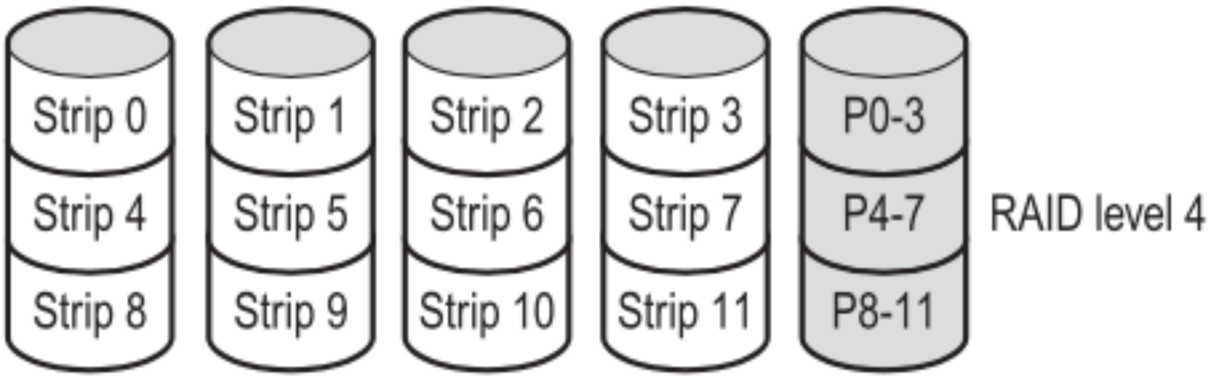


RAID 2

Problem 5.22 (7/9)

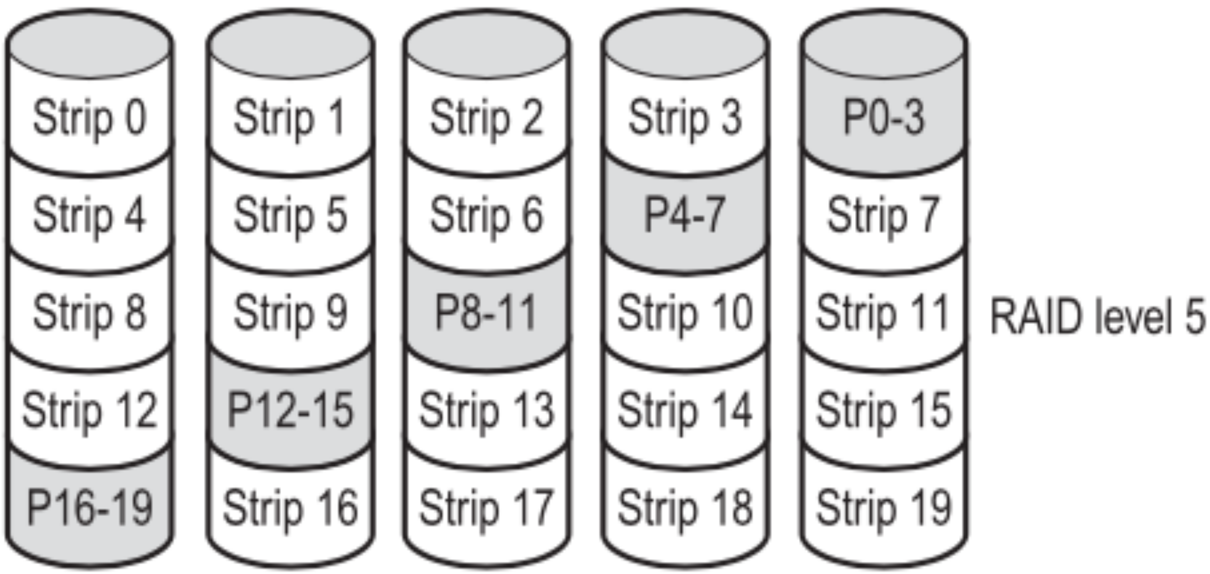
RAID 3

Problem 5.22 (8/9)



RAID 4

Problem 5.22 (9/9)



RAID 5

Problem 5.22 – Solution (1/4)

RAID	Read Performance
0	Parallel reads for **one read request**
1	Two parallel reads for **two read requests**
2	Parallel reads for **one read request**
3	Parallel reads for **one read request**
4	Parallel reads for **one read request**
5	Parallel reads for **one read request**

Problem 5.22 – Solution (2/4)

RAID	Write Performance
0	Normal single disk performance
1	Normal single disk performance
2	Reduced performance; cannot serve multiple requests simultaneously; good for writing large files
3	The worst performance for small files; the good performance for writing large sequential data
4	Low performance due to usage only one disk for parity
5	Reduced performance (RAID 0 < RAID 5 < RAID 4)

Problem 5.22 – Solution (3/4)

RAID	Space Overhead
0	0 %
1	100 %
2	With 32-bit word and 6 parity bits (as described in TB14) the overhead is ~19%
3	With 32-bit word and 1 parity bit the overhead is ~3%
4	Same as in RAID 3
5	Same as in RAID 3

Problem 5.22 – Solution (4/4)

RAID	Reliability
0	No reliability
1	Can survive one disk crash
2	Can survive one disk crash; a single random bit error in a word can be detected and corrected
3	Can survive one disk crash; a single random bit error in a word can be detected
4	Can survive one disk crash; a single random bit error in a word can be detected
5	Can survive one disk crash; a single random bit error in a word can be detected

Problem 5.31

- Disk requests come in to the disk driver for cylinders 10, 22, 20, 2, 40, 6, and 38, in that order. A seek takes 6 msec per cylinder. How much seek time is needed for:
    A. First-come, first served
    B. Closest cylinder next
    C. Elevator algorithm (initially moving upward)
- In all cases, the arm is initially at cylinder 20

Problem 5.31 – Solution (1/3)

A. First-come, first served (10, 22, 20, 2, 40, 6, 38):

- First request: arm is initially at cylinder 20, moves to cylinder 10: it travels 10 cylinders
- Second request: 10 –> 22 = 12 cylinders
- Third request: 22 -> 20 = 2 cylinders
- Fourth request: 20 -> 2 = 18 cylinders
- Fifth request: 2 –> 40 = 38 cylinders
- Sixth request: 40 -> 6 = 34 cylinders
- Seventh request: 6 -> 38 = 32 cylinders
- Total: 10 + 12 + 2 + 18 + 38 + 34 + 32 = 146 cylinders
- 146 * 6 msec = 876 msec

Problem 5.31 – Solution (2/3)

B. Closest cylinder next (10, 22, 20, 2, 40, 6, 38):

- 20 -> 20 = 0 cylinders
- 20 –> 22 = 2 cylinders
- 22 -> 10 = 12 cylinders
- 10 -> 6 = 4 cylinders
- 6 –> 2 = 4 cylinders
- 2 -> 38 = 36 cylinders
- 38 -> 40 = 2 cylinders
- Total: 2 + 12 + 4 + 4 + 36 + 2 = 60 cylinders
- 60 * 6 msec = 360 msec

Problem 5.31 – Solution (3/3)

C. Elevator algorithm (10, 22, 20, 2, 40, 6, 38):

- 20 -> 20 = 0 cylinders
- 20 –> 22 = 2 cylinders
- 22 -> 38 = 16 cylinders
- 38 -> 40 = 2 cylinders
- 40 –> 10 = 30 cylinders
- 10 -> 6 = 4 cylinders
- 6 -> 2 = 4 cylinders
- Total: 2 + 16 + 2 + 30 + 4 + 4 = 58 cylinders
- 58 * 6 msec = 348 msec

Problem 5.39 (1/2)

- A system simulates multiple clocks by chaining all pending clock requests together as shown in Fig. 5-30
- Suppose the current time is 5000 and there are pending clock requests for time 5008, 5012, 5015, 5029, and 5037
- Show the values of Clock header, Current time, and Next signal at times 5000, 5005, and 5013
- Suppose a new (pending) signal arrives at time 5017 for 5033. Show the values of Clock header, Current time and Next signal at time 5023

Problem 5.39 (2/2)

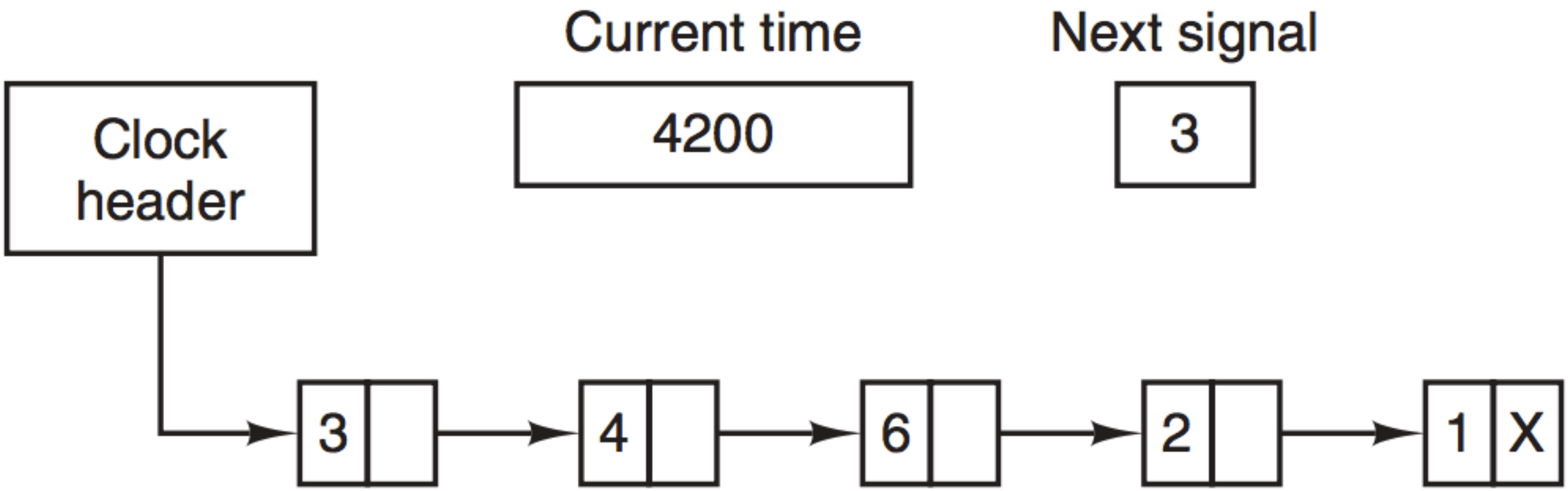


Figure 5-30. Simulating multiple timers with a single clock

Problem 5.39 – Solution (1/4)

- The first value of the header is the difference between the current time and pending time. The next values are differences between following times. For example, at time 5000:
  - Current time = 5000
  - Next Signal = 8
  - Header → 8 → 4 → 3 → 14 → 8

Problem 5.39 – Solution (2/4)

- At time 5005 (pending requests at 5008, 5012, 5015, 5029, 5037):
  - Current time = 5005
  - Next Signal = 3
  - Header → 3 → 4 → 3 → 14 → 8

Problem 5.39 – Solution (3/4)

- At time 5013 (pending requests at 5015, 5029, 5037):
  - Current time = 5013
  - Next Signal = 2
  - Header 2 → 14 → 8

Problem 5.39 – Solution (4/4)

- At time 5023 (pending requests at 5029, 5033, 5037):
  - Current time = 5023
  - Next Signal = 6
  - Header → 6 → 4 → 4

Problem 5.44

- The designers of a computer system expected that the mouse could be moved at a maximum rate of 20 cm/sec. If a mickey is 0.1 mm and each mouse message is 3 bytes, what is the maximum data rate of the mouse assuming that each mickey is reported separately?

Problem 5.44 – Solution

- The maximum rate the mouse can move is 200 mm/sec, which is 2000 mickeys/sec.
- If each report is 3 byte, the output rate is 6000 bytes/sec

Problem 5.47

- Assuming that it takes 2 nsec to copy a byte, how much time does it take to completely rewrite the screen of an 80 character × 25 line text mode memory-mapped screen?
- What about a 1024 × 768 pixel graphics screen with 24-bit color?

Problem 5.47 – Solution

- Rewriting the text screen requires copying 2000 bytes, which can be done in 4 µseconds. Rewriting the graphics screen requires copying $1024 \times 768 \times 3 = 2{,}359{,}296$ bytes, or about 4.72 msec.

Problem 5.52

- Describe two advantages and two disadvantages of thin client computing

Problem 5.52 – Solution (1/2)

- Advantages:
  - Low cost
  - No need for complex management for the clients

Problem 5.52 – Solution (2/2)

- Disadvantages:
  - Lower performance due to network latency
  - Potential loss of privacy (the client's data/ information is shared with the server)

Problem 5.53

- If a CPU's maximum voltage *V* is cut to *V/n*, its power consumption drops to $1/n^2$ of its original value and its clock speed drops to $1/n$ of its original value. Suppose that a user is typing at 1 char/sec, but the CPU time required to process each character is 100 msec. What is the optimal value of *n* and what is the corresponding energy saving in percent compared to not cutting the voltage? Assume that an idle CPU consumes no energy at all

Problem 5.53 – Solution

- If n = 10, the CPU can still get its work done on time, but the energy used drops considerably
- If the energy consumed in 1 sec at full speed is E, then running at full speed for 100 msec then going idle for 900 msec uses $E/10$
- Running at 1/10 speed for a whole second uses $E/100$, a saving of $9E/100$
- The percent savings by cutting the voltage is 90%

End

Week 12 – Tutorial