Griffith UNIVERSITY

2803ICT

I/O Management, Disk Scheduling

based on
William Stallings
Chapter 11

TOPICS

1) Introduction and overview
2) Advanced C programming
3) Memory management
4) File system and device IO
5) Socket programming
6) Processes
7) Interprocess communication (IPC)
8) Multithreading
9) Synchronisation (*2 weeks*)
10) Distributed software

Categories of I/O Devices

- External devices that engage in I/O with computer systems can be grouped into three categories:

1. **Human readable**

   suitable for communicating with the computer user
   printers, terminals, video display, keyboard, mouse

2. **Machine readable**

   suitable for communicating with electronic equipment
   disk drives, USB keys, sensors, controllers

3. **Communication**

   suitable for communicating with remote devices
   modems, digital line drivers

Data Rates

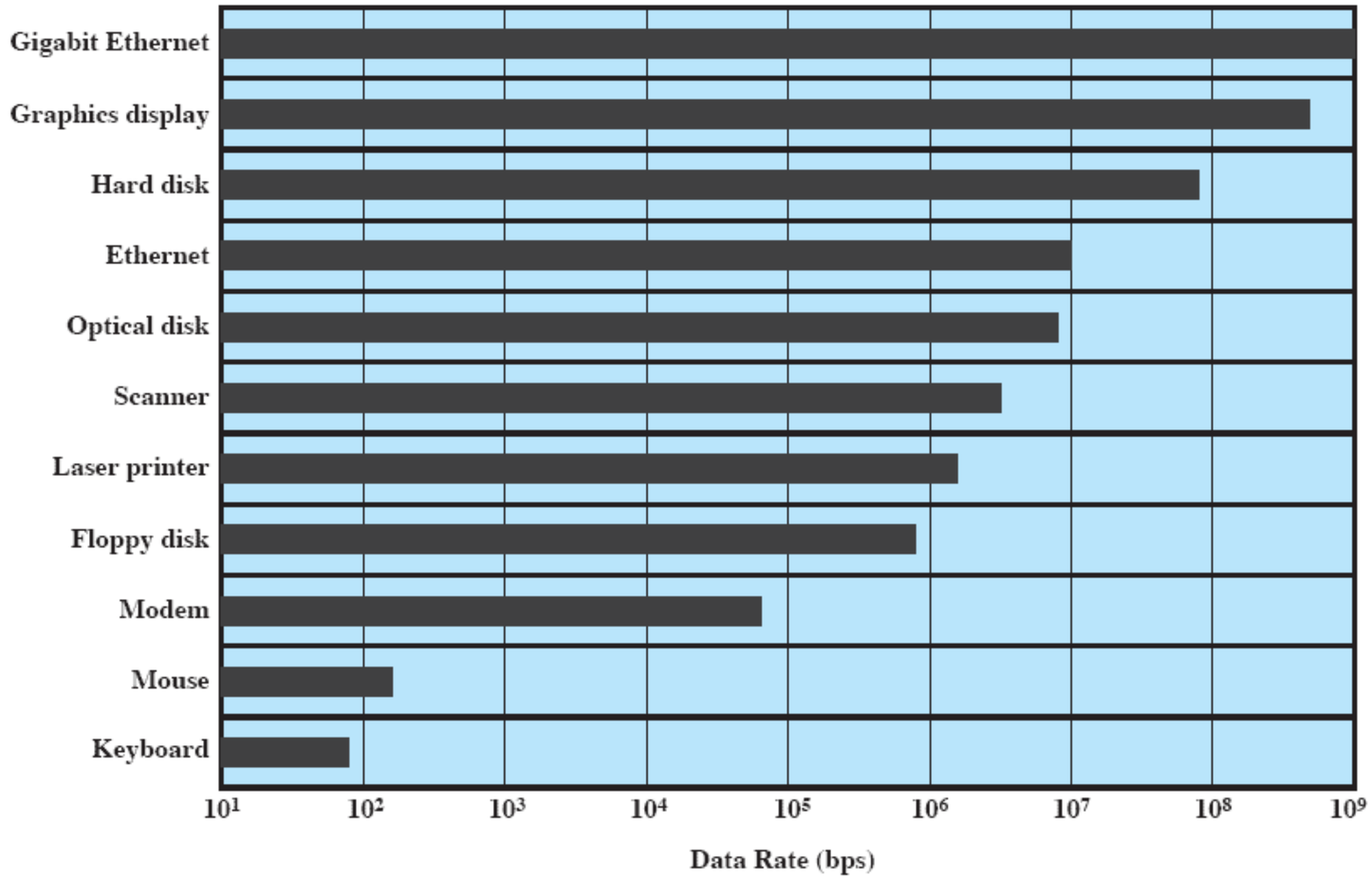


Figure 11.1 Typical I/O Device Data Rates

I/O Techniques

- **3 techniques for performing I/O are:**
- **Programmed I/O**
  - the processor issues an I/O command on behalf of a process to an I/O module; that process then busy waits for the operation to be completed before proceeding
- **Interrupt-driven I/O**
  - the processor issues an I/O command on behalf of a process
  - if non-blocking – processor continues to execute instructions from the process that issued the I/O command
  - if blocking – the next instruction the processor executes is from the OS, which will put the current process in a blocked state and schedule another process
- **Direct Memory Access (DMA)**
  - a DMA module controls the exchange of data between main memory and an I/O modul

Techniques for Performing I/O

Griffith UNIVERSITY

Table 11.1 I/O Techniques

	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

Evolution of the I/O Function

1. Processor directly controls a peripheral device
2. A controller or I/O module is added
3. Same configuration as step 2, but now interrupts are employed
4. The I/O module is given direct control of memory via DMA
5. The I/O module is enhanced to become a separate processor, with a specialised instruction set tailored for I/O
6. The I/O module has a local memory of its own and is, in fact, a computer in its own right

Direct Memory Access

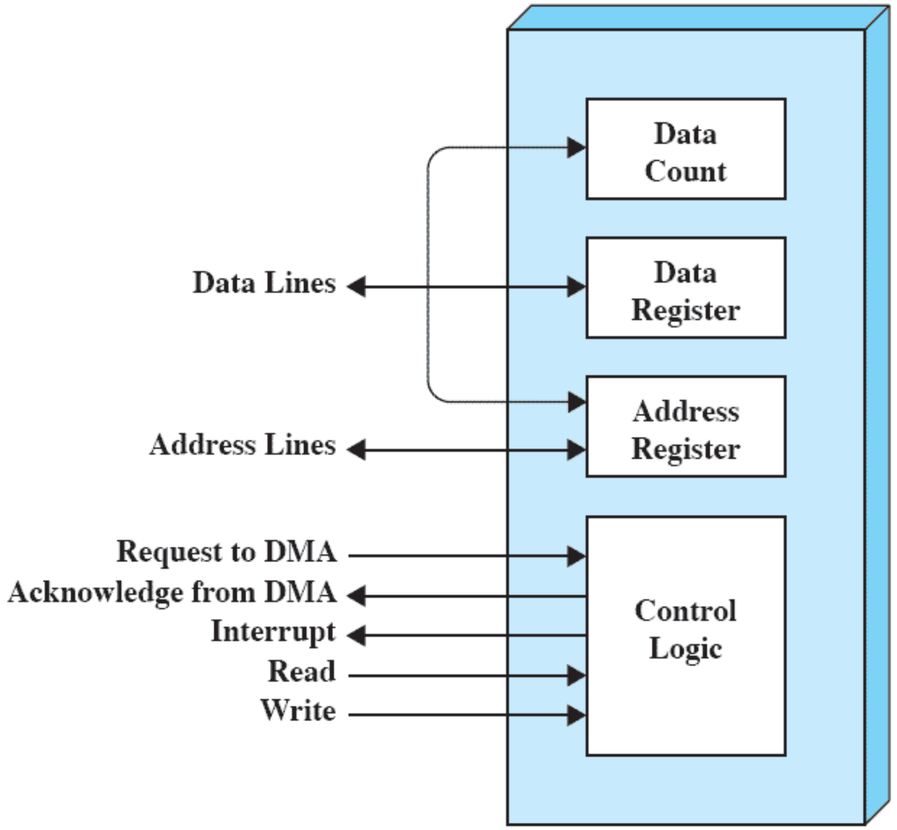
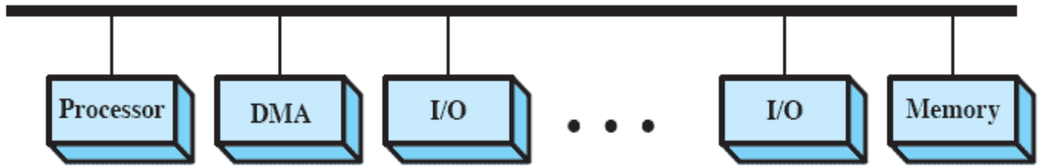
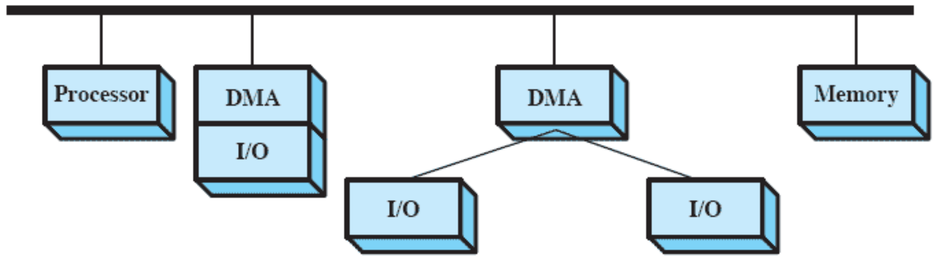


Figure 11.2 Typical DMA Block Diagram

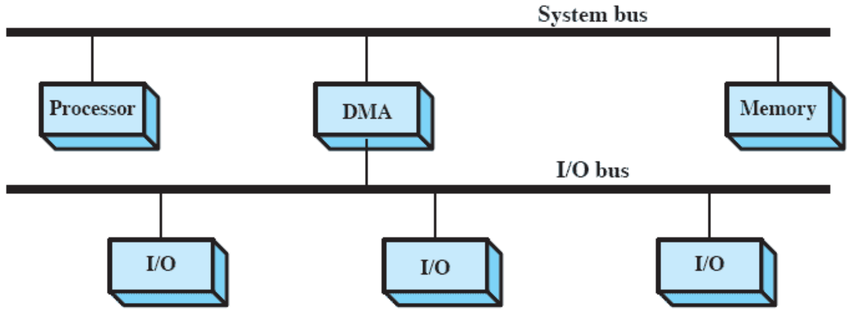
DMA Alternative Configurations



(a) Single-bus, detached DMA



(b) Single-bus, Integrated DMA-I/O



(c) I/O bus

Griffith UNIVERSITY

2803ICT

I/O Management, Disk Scheduling

Part 2

based on
William Stallings
Chapter 11

Design Objectives

- **Efficiency**
- Major effort in I/O design
- Important because I/O operations often form a bottleneck
- Most I/O devices are extremely slow compared with main memory and the processor
- The area that has received the most attention is disk I/O

- **Generality**
- Desirable to handle all devices in a uniform manner
- Applies to the way processes view I/O devices and the way the operating system manages I/O devices and operations
- Diversity of devices makes it difficult to achieve true generality
- Use a hierarchical, modular approach to the design of the I/O function

Hierarchical Design

- Functions of the operating system should be separated according to their complexity, their characteristic time scale, and their level of abstraction
- Leads to an organisation of the operating system into a series of layers
- Each layer performs a related subset of the functions required of the operating system
- Layers should be defined so that changes in one layer do not require changes in other layers

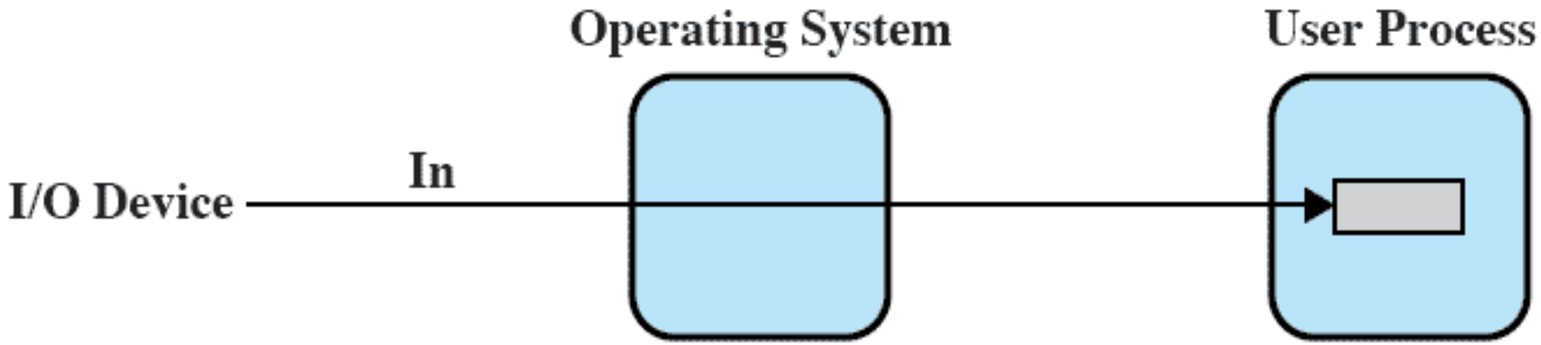
Buffering

- Perform input transfers in advance of requests being made and perform output transfers some time after the request is made

Block-oriented device	Stream-oriented device
• stores information in blocks that are usually of **fixed size** • transfers are made one block at a time • possible to reference data by its block number • disks and **USB keys** are examples	• transfers data in and out as a **stream** of bytes • no block structure • terminals, printers, communications ports, and most other devices that are not secondary storage are examples

No Buffer

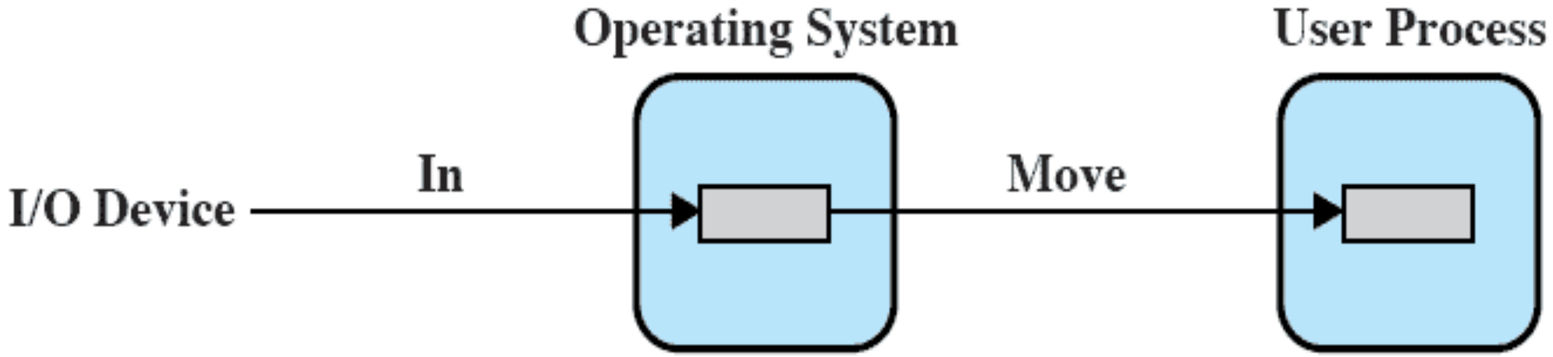
- Without a buffer, the OS directly accesses the device when it need



(a) No buffering

Single Buffer

- Operating system assigns a buffer in main memory for an I/O request



(b) Single buffering

Block-Oriented Single Buffer

- Input transfers are made to the system buffer
- Reading ahead/anticipated input
    - is done in the expectation that the block will eventually be needed
    - when the transfer is complete, the process moves the block into user space and immediately requests another block
- Generally provides a **speedup** compared to the lack of system buffering
- **Disadvantages**:
    - complicates the logic in the operating system
    - swapping logic is also affected

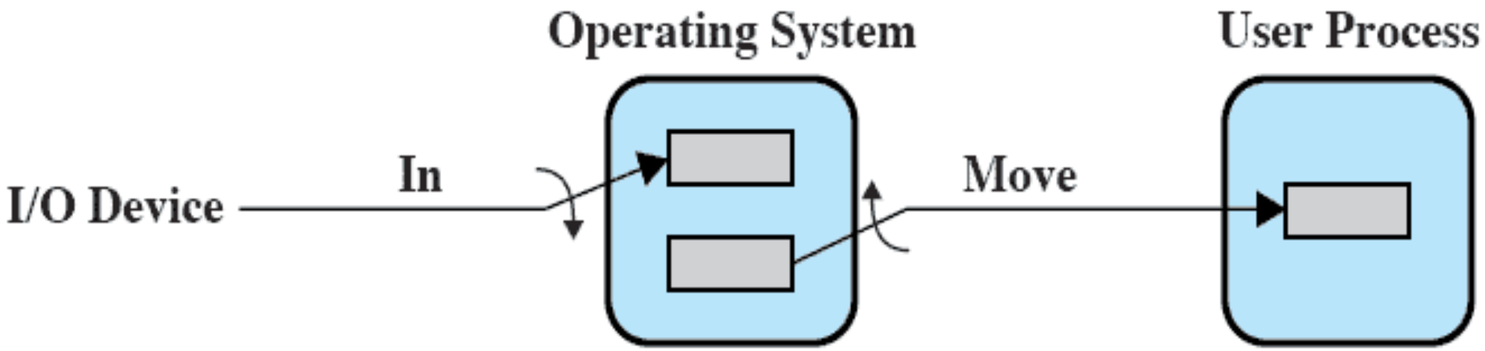
Stream-Oriented Single Buffer

- **Line-at-a-time operation**
  - appropriate for scroll-mode terminals (dumb terminals)
  - user input is one line at a time with a carriage return signalling the end of a line
  - output to the terminal is similarly one line at a time

- **Byte-at-a-time operation**
  - used on forms-mode terminals
  - when each keystroke is significant
  - other peripherals such as sensors and controllers

Double Buffer

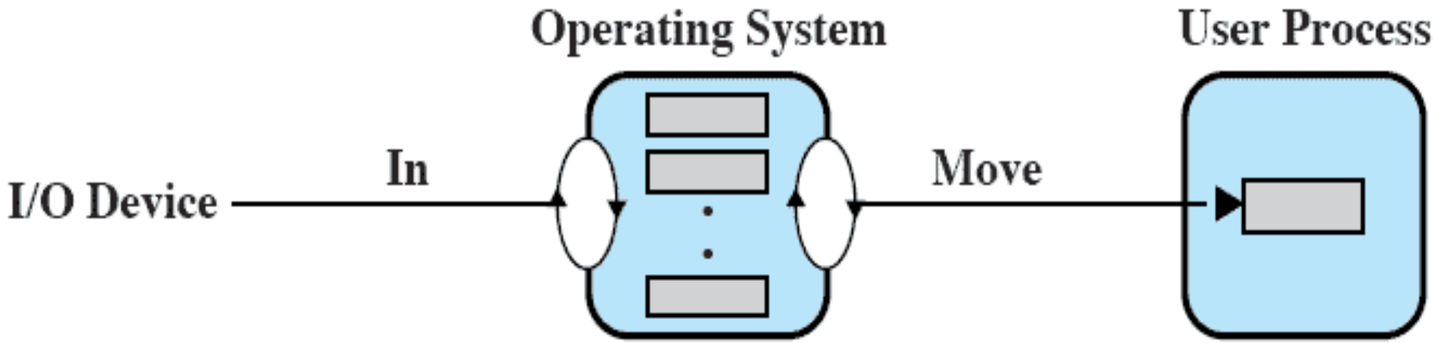
- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer
- Also known as buffer swapping



(c) Double buffering

Circular Buffer

- Two or more buffers are used
- Each individual buffer is one unit in a circular buffer
- Used when I/O operation must **keep up with process**



(d) Circular buffering

The Utility of Buffering

- Technique that **smoothes out peaks** in I/O demand
    - with enough demand eventually all buffers become full and their advantage is lost
- When there is a variety of I/O and process activities to service, buffering can increase the efficiency of the OS and the performance of individual processes

Griffith UNIVERSITY

2803ICT

I/O Management, Disk Scheduling

Part 3

based on
William Stallings
Chapter 11

Disk Performance Parameters

- The actual details of disk I/O operation depend on the:
  - computer system
  - operating system
  - nature of the I/O channel and disk controller hardware

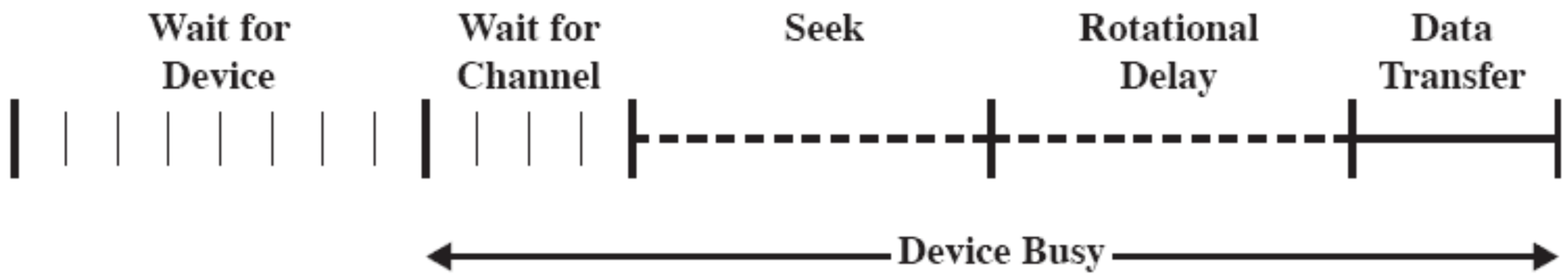
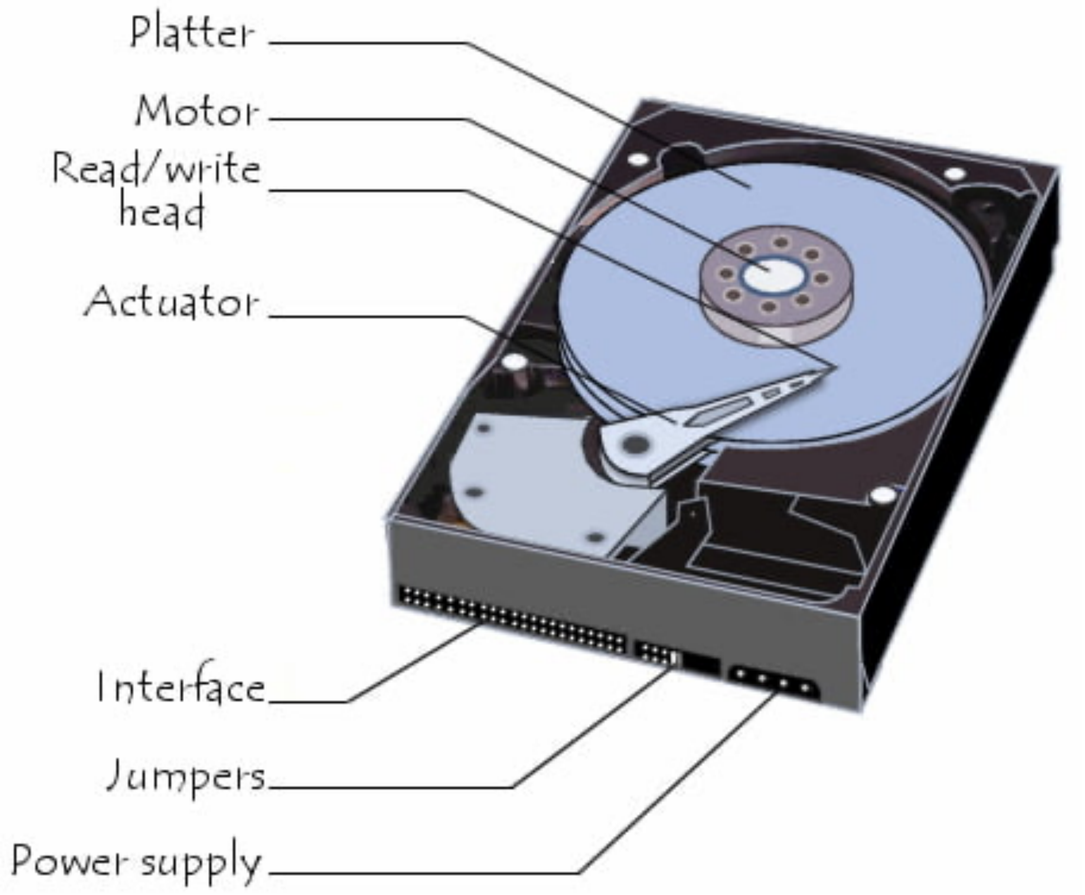
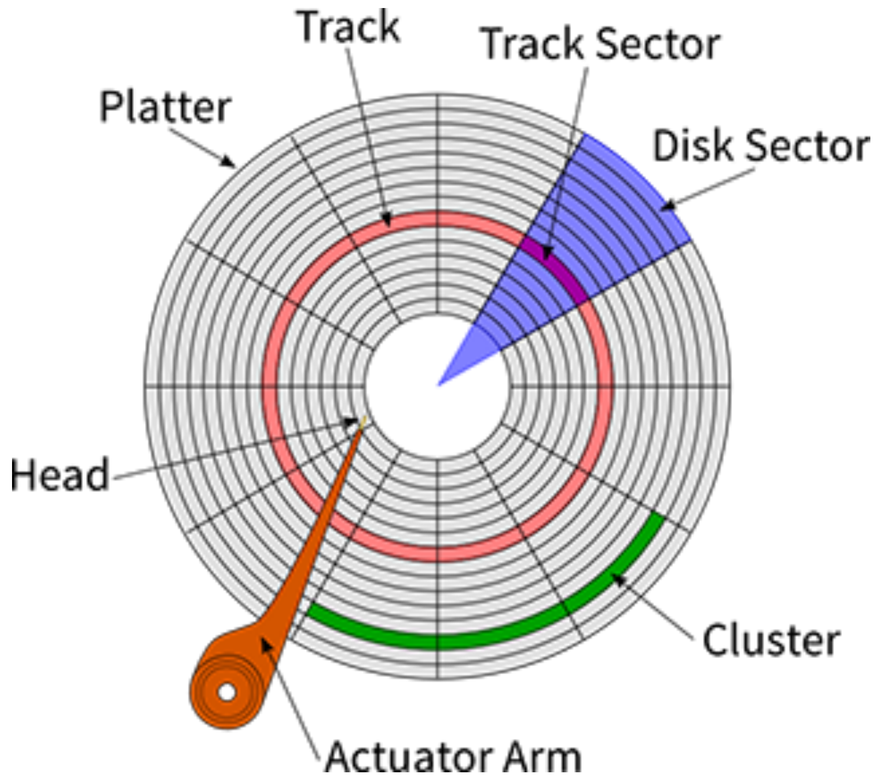


Figure 11.6 Timing of a Disk I/O Transfer

Disk Performance Parameters





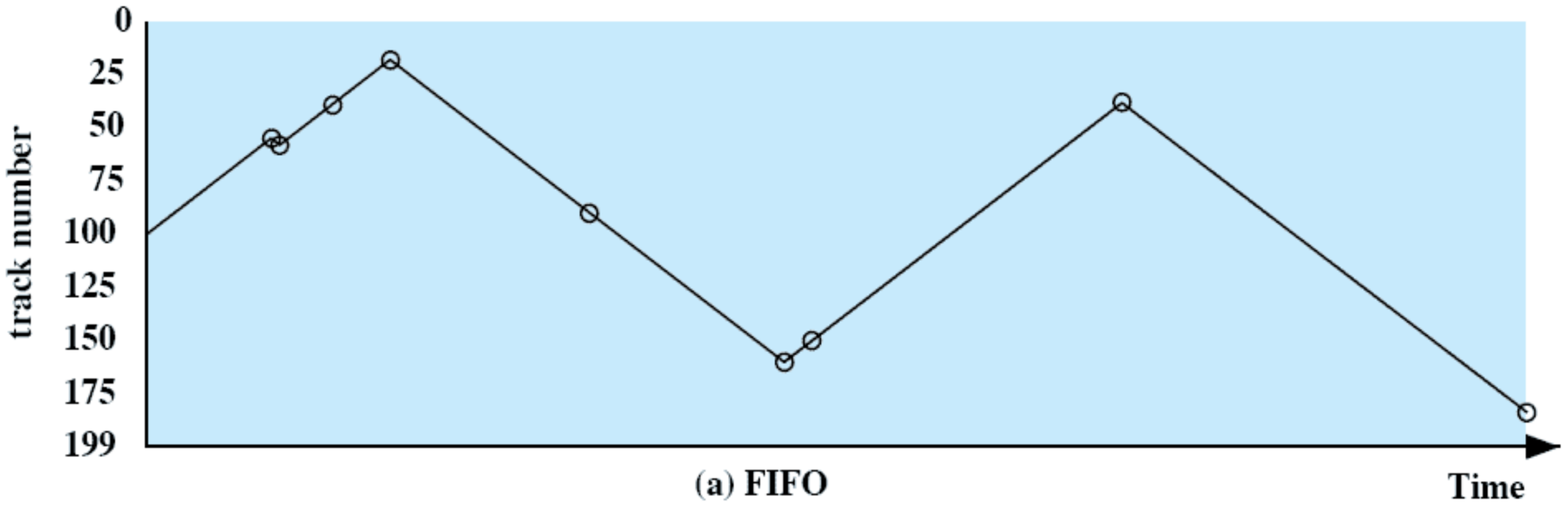
Positioning the Read/Write Heads

Griffith UNIVERSITY

- When the disk drive is operating, the disk is **rotating at constant speed**
- To read or write the **head must be positioned** at the desired track and at the beginning of the desired sector on that track
- Track selection involves **moving the head** in a movable-head system or electronically selecting one head on a fixed-head system
- On a movable-head system the time it takes to position the head at the track is known as **seek time**
- The time it takes for the beginning of the sector to reach the head is known as **rotational delay**
- The sum of the seek time and the rotational delay equals the **access time**

First In First Out (FIFO)

- Processes in sequential order
- Fair to all processes
- **Approximates random scheduling** in performance if there are many processes competing for the disk

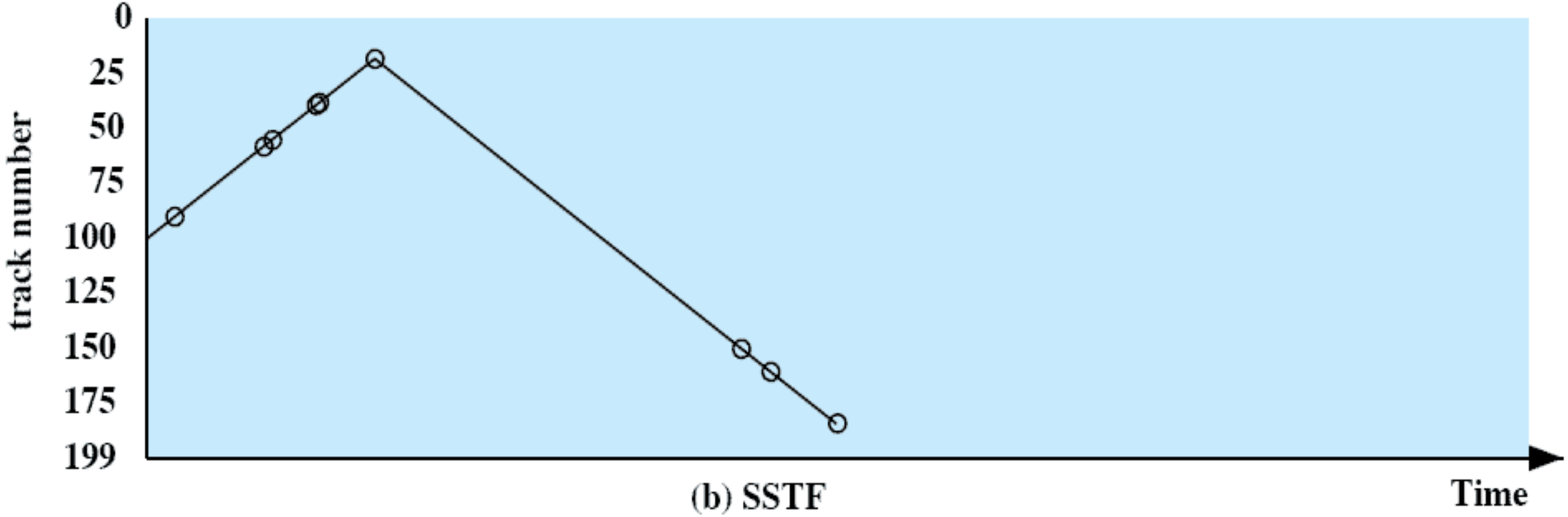


Priority (PRI)

- Control of the scheduling is outside the control of disk management software
- Goal is not to optimise disk utilisation but to **meet other objectives**
- **Short** batch jobs and **interactive** jobs are given higher priority
- Provides good **interactive response time**
- Longer jobs may have to **wait an excessively long time**
- A **poor policy for database systems**

Shortest Service Time First (SSTF)

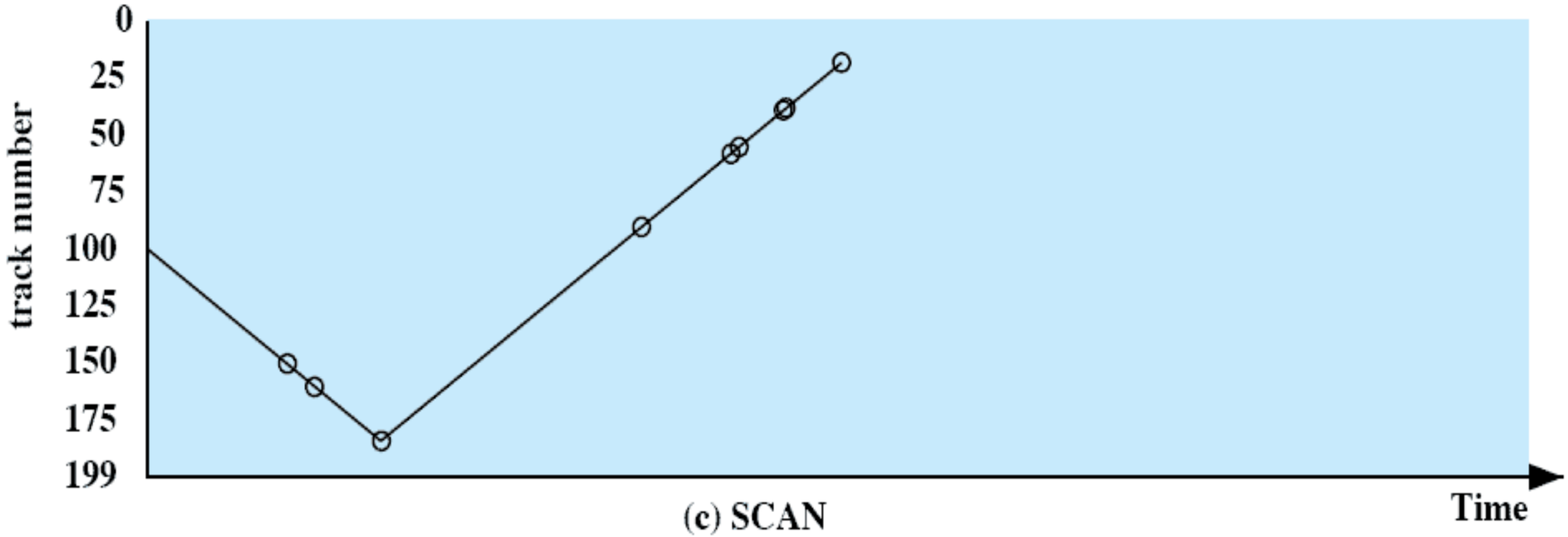
- Select the disk I/O request that requires **the least movement of the disk arm** from its current position
- Always choose the **minimum seek time**



(b) SSTF

SCAN

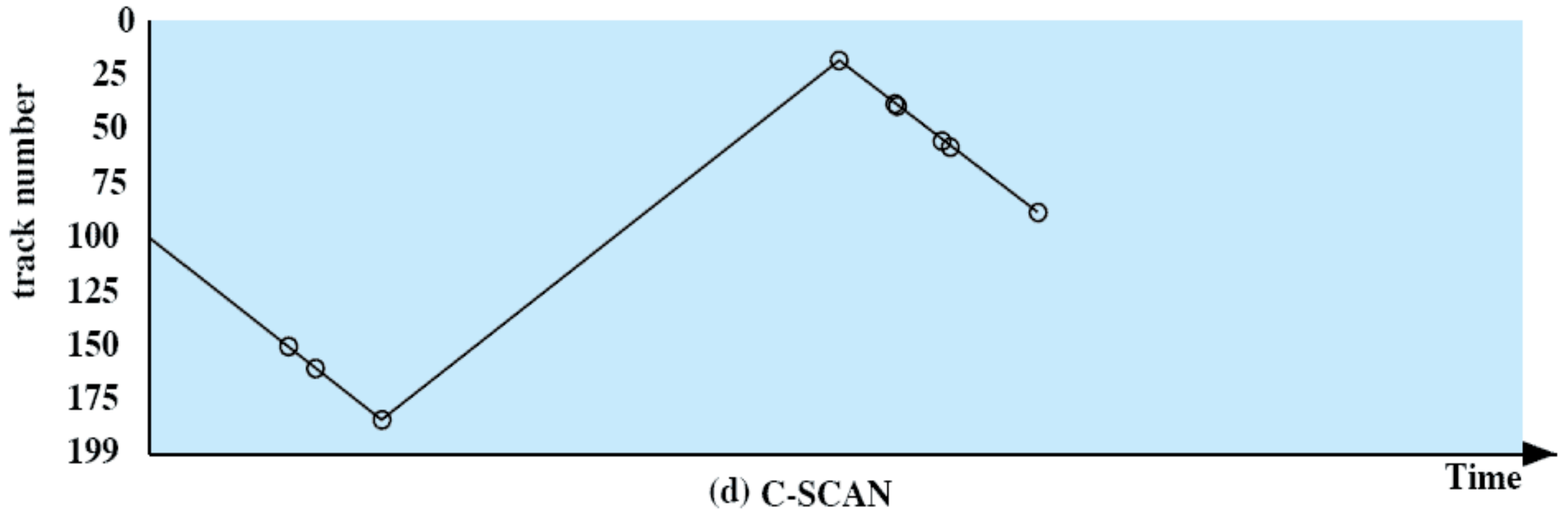
- Also known as the **elevator algorithm**
- Arm moves in **one direction** only
  - satisfies all outstanding requests until it reaches the last track in that direction then the **direction is reversed**
- Favours jobs whose requests are for tracks nearest to both innermost and outermost tracks



(c) SCAN

C-SCAN (Circular SCAN)

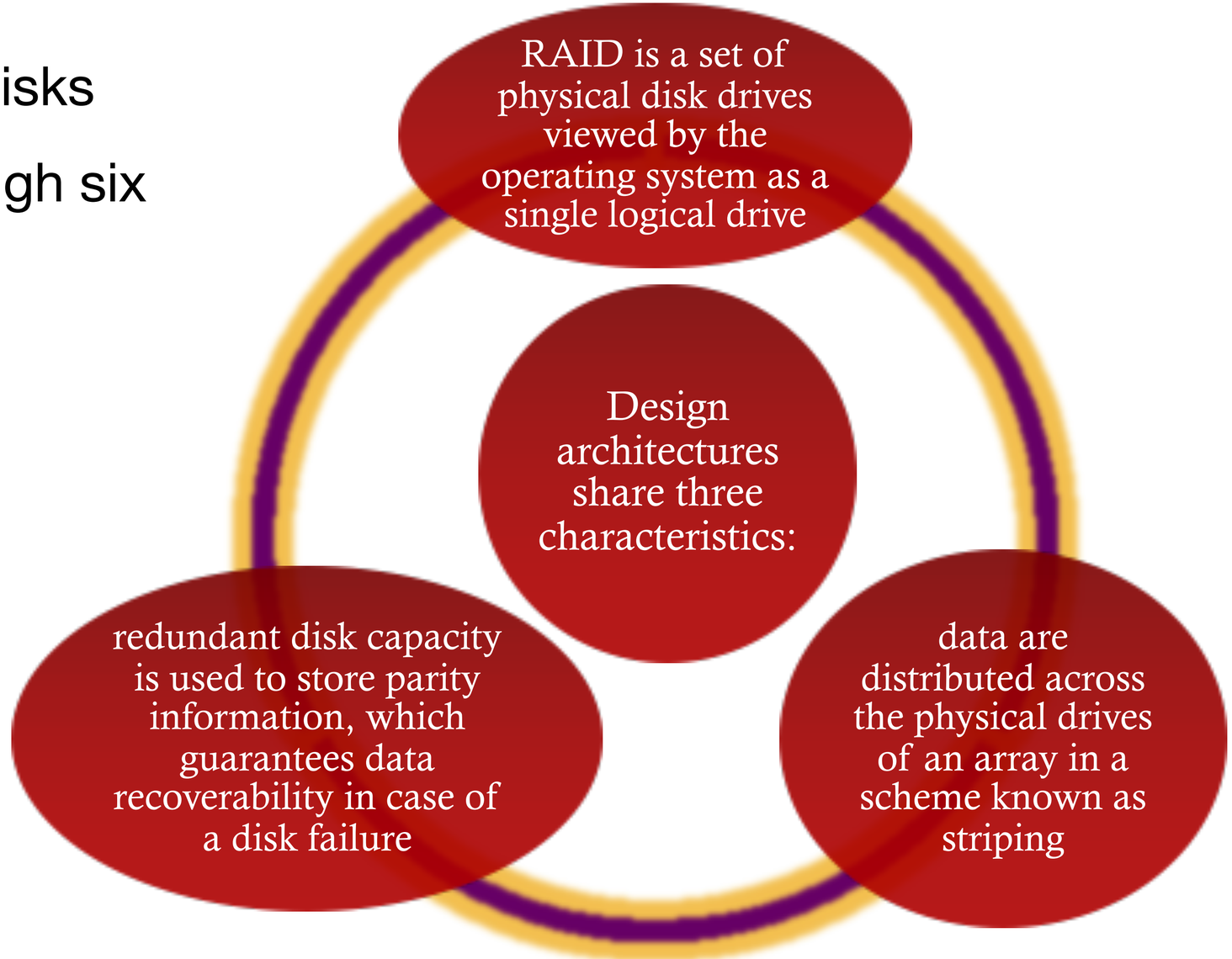
- Restricts scanning to **one direction only**
- When the last track has been visited in one direction, the arm is **returned to the opposite end** of the disk and the scan begins again



(d) C-SCAN

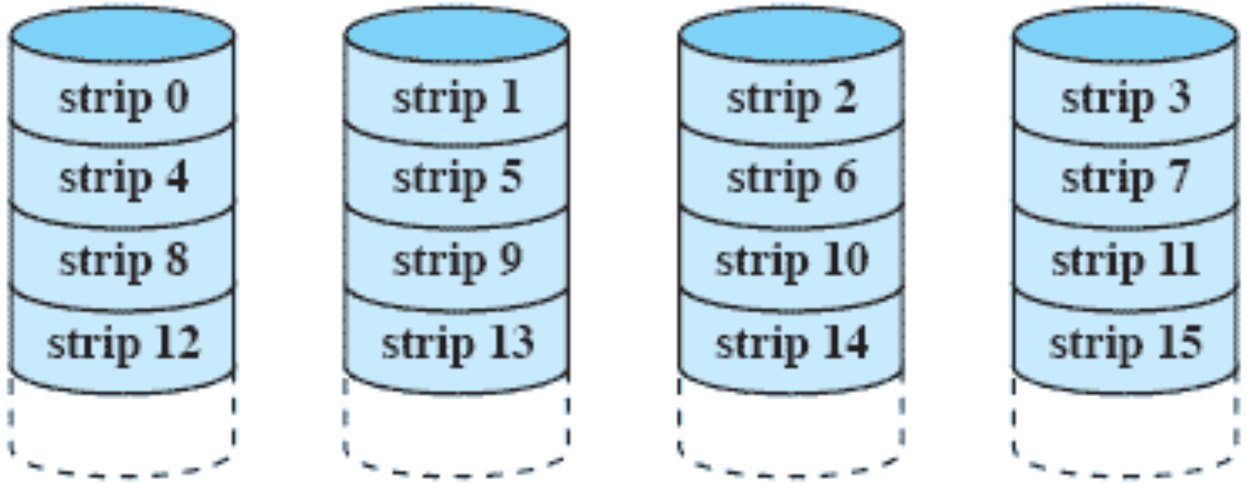
RAID

- **R**edundant **A**rray of **I**ndependent **D**isks
- Consists of seven levels, zero through six



RAID - Level 0

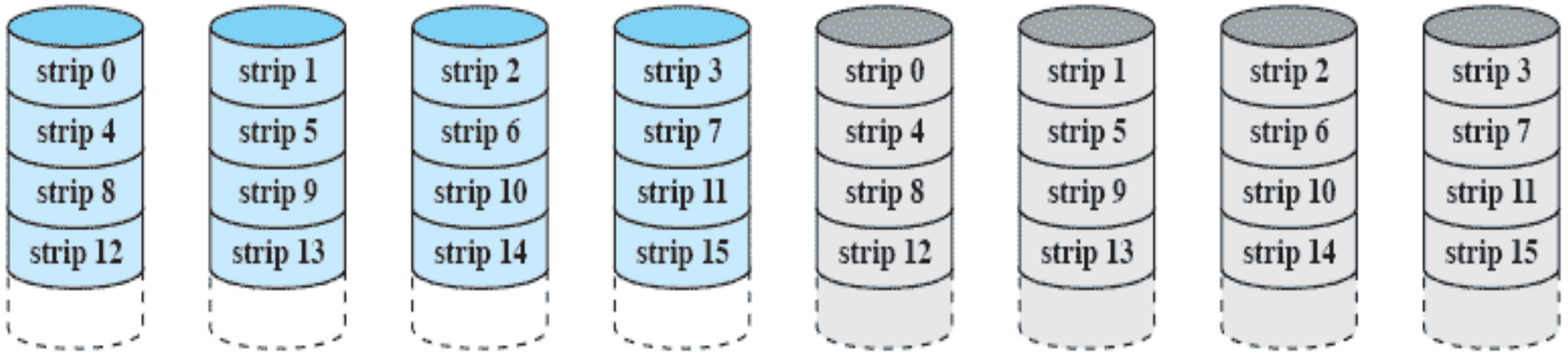
- **Not a true RAID** because it does not include redundancy to improve performance or provide data protection
- User and system data are distributed across all of the disks in the array
- Logical disk is divided into strips



(a) RAID 0 (non-redundant)

RAID - Level 1

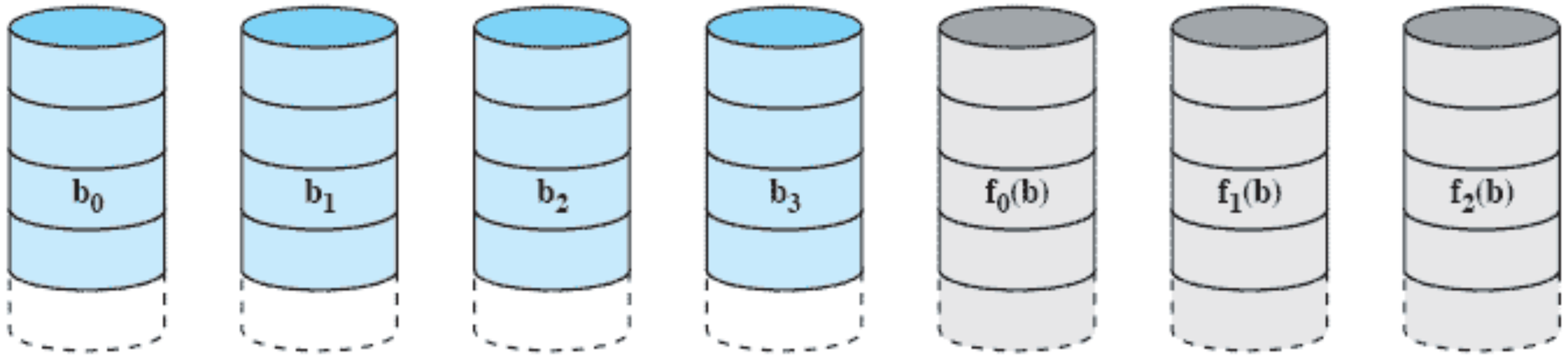
- Redundancy is achieved by the simple expedient of **duplicating all the data**
- There is no “write penalty”
- When a drive fails the data may still be accessed from the second drive
- Principal **disadvantage is the cost**



(b) RAID 1 (mirrored)

RAID - Level 2

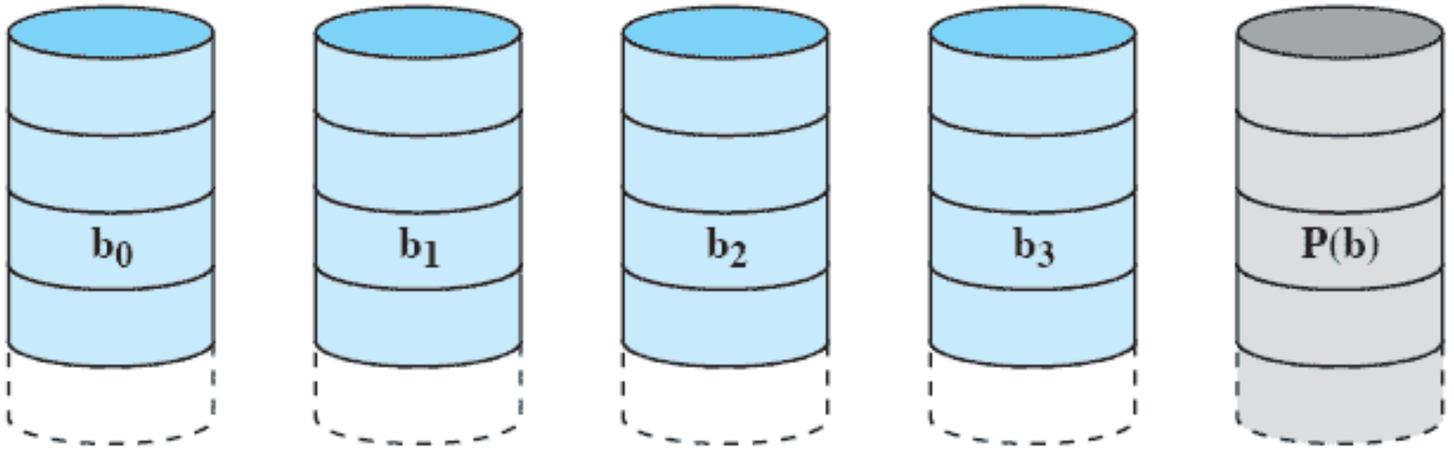
- Makes use of a parallel access technique
- Data **striping** is used
- Typically a **Hamming code** is used
- Effective choice in an environment in which many disk errors occur



(c) RAID 2 (redundancy through Hamming code)

RAID - Level 3

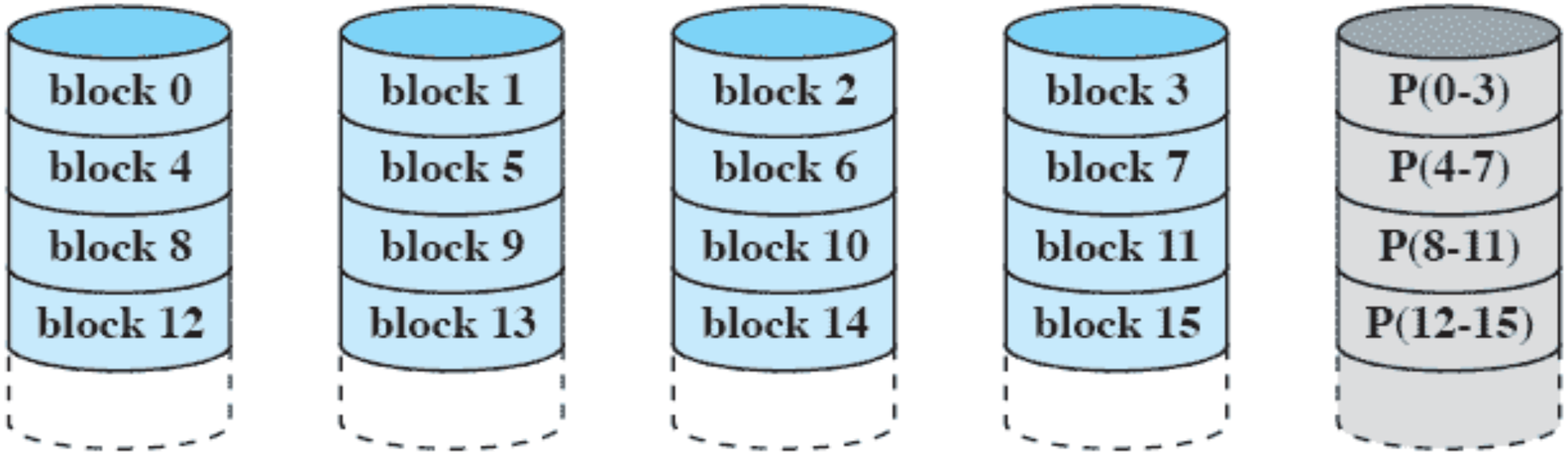
- Requires only a **single redundant disk**, no matter how large the disk array
- Employs parallel access, with data distributed in small strips
- Can achieve **very high data transfer rates**



(d) RAID 3 (bit-interleaved parity)

RAID - Level 4

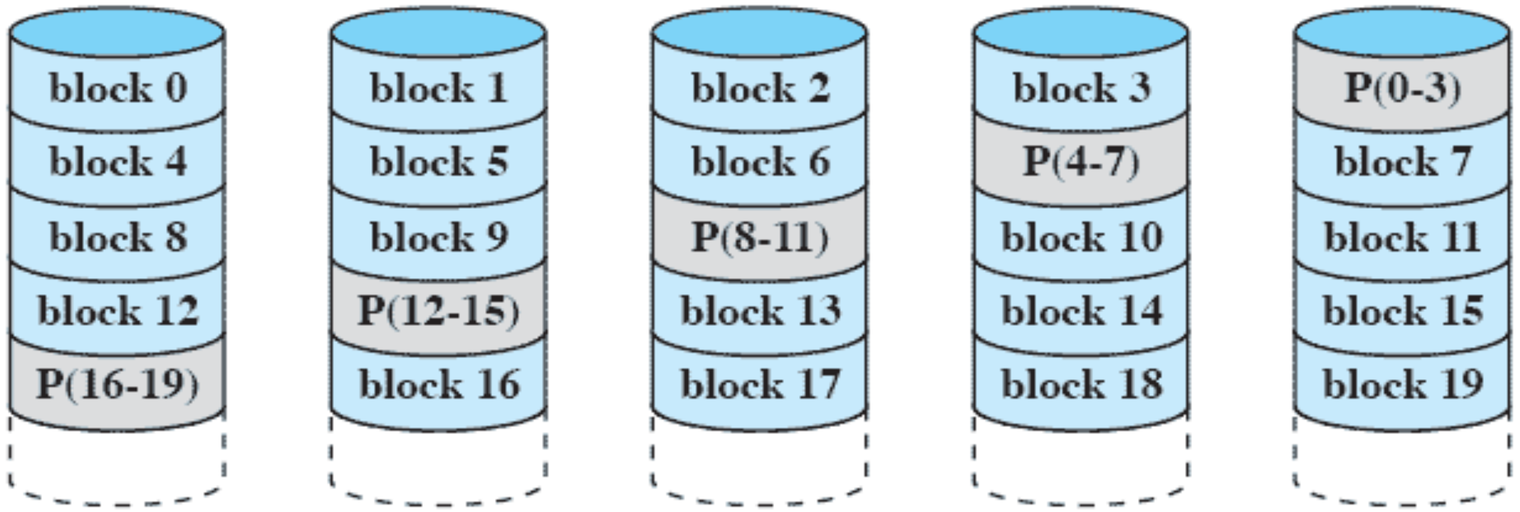
- Makes use of an independent access technique
- A bit-by-bit parity strip is calculated across corresponding strips on each data disk, and the parity bits are stored in the corresponding strip on the parity disk
- Involves a write penalty when an I/O write request of small size is performed



(e) RAID 4 (block-level parity)

RAID - Level 5

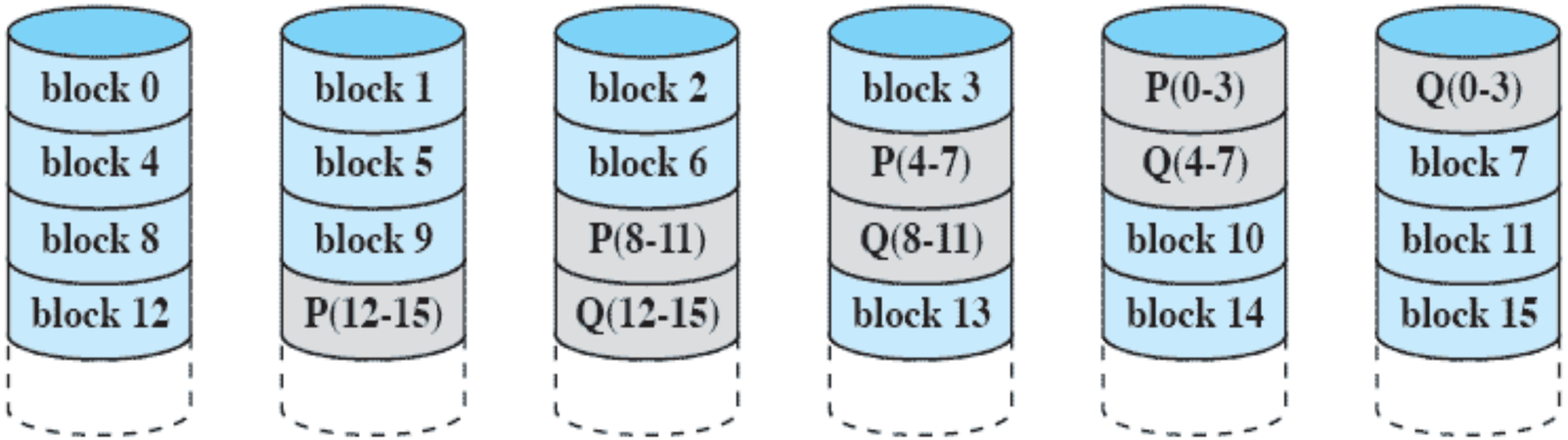
- Similar to RAID-4 but **distributes the parity bits across all disks**
- Typical allocation is a **round-robin** scheme
- Has the characteristic that the **loss of any one disk does not result in data loss**



(f) RAID 5 (block-level distributed parity)

RAID - Level 6

- **Two different parity calculations** are carried out and stored in separate blocks on different disks
- Provides **extremely high data availability**
- **Protects against spare exhaustion** during a rebuild of the array after one disk failed.



(g) RAID 6 (dual redundancy)

Solid State Disks (SSD)

- Have **no moving parts**
- Electro-static storage of bit information
    - Reading equivalent to measuring the charge of a capacitor
    - Writing requires discharging (0 bit) and then possibly re-applying a charge (e.g. 1 bit)
- Data is stored in Flash Memory
    - Quick discharge of whole blocks of data
    - Changing a single bit requires clearing and re-writing whole block
- Read operations much faster than write operations

SSD Characteristics

- Extremely Fast access speed
  - 100,000 IOPS (Server SSDs: 1.2 MIOPS)
  - Factor 250x over ultra-fast Hard Disks (max. 400 IOPS)
- Fast read burst speed
  - Recent SSDs: 300-1200 MB/s (Server SSDs: up to 5000 MB/s)
  - Can saturate SATA 6G: 500 MB/s
- Reasonable write speeds
  - Recent SSDs: 250-450 MB/s (Server SSDs: up to 2000 MB/s)

SLC vs MLC

- **SLC**
    - Single-Level Cells
    - Extremely fast write speeds
    - 100,000 write cycles or more
    - Expensive
- **MLC**
    - Multi-Level Cell
    - Voltage level of charge determines bit combination
    - Slower write access
    - Appx 5000 write cycles
    - Cheaper to produce

SSD Access Strategies

- Operating System or SSD Controller
- Wear-Leveling
    - Re-arrange Blocks on Write
    - Attempt to spread around write operations equally across all blocks
    - Requires block mapping data structures
- Spare Blocks
    - Replace unusable blocks to increase life span of drive
    - Up to 30% of the capacity of modern SSD
- Disabling of Defragmentation
    - Fragmentation matters less on SSDs: no seek time
    - Can be counter-productive (significant number of additional write ops)