THE UNIVERSITY OF
NEWCASTLE
AUSTRALIA

FACULTY OF
ENGINEERING AND
BUILT ENVIRONMENT

www.newcastle.edu.au

OPERATING SYSTEMS

Week 9

Much of the material on these slides comes from the recommended textbook by William Stallings

Detailed content

Weekly program

- ✓ Week 1 – Operating System Overview
- ✓ Week 2 – Processes and Threads
- ✓ Week 3 – Scheduling
- ✓ Week 4 – Real-time System Scheduling and Multiprocessor Scheduling
- ✓ Week 5 – Concurrency: Mutual Exclusion and Synchronization
- ✓ Week 6 – Concurrency: Deadlock and Starvation
- ✓ Week 7 – Memory Management
- ✓ Week 8 – Memory Management II
- ➡ ❑ **Week 9 – Disk and I/O Scheduling**
- ❑ Week 10 – File Management
- ❑ Week 11 – Security and Protection
- ❑ Week 12 – Revision of the course
- ❑ Week 13 – Extra revision (if needed)

Key Concepts From Last Lecture

- A Number of design issues relate to OS support for memory management:
- **The OS must provide**
  - a policy for **fetching** a page into main memory: *demand*, or *pre-paging*
  - a policy for **placing** a page in memory
  - a policy for **replacing** a (or swapping out) a page in memory; there are several methods: optimal, LRU, FIFO, clock
  - a **resident set management policy**:
  - a **cleaning policy**: when should a modified page be written to disk?
    - only when replaced? or as soon as they are modified?
  - a **load control mechanism** for controlling the amount of page faults.

Week 09 Lecture Outline

Disk and I/O Scheduling

- □ I/O device characteristics
- □ Organization of I/O functions
- □ DMA
- □ OS design issues
- □ Buffering
- □ Disk Scheduling algorithms
- □ RAID architecture

Videos to watch before lecture

I/O

- The messiest aspect of OS design
- Wide variety of I/O devices
  - Wide variation in performance
- Difficult to provide a general, consistent solution

Categories of I/O devices

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

Human readable

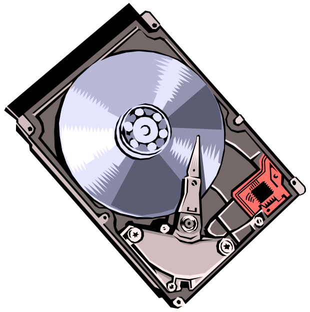
- suitable for communicating with the computer user
- printers, terminals, video display, keyboard, mouse

Machine readable

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

Communication

- suitable for communicating with remote devices
- modems, digital line drivers



Difference in I/O devices

- Devices differ in a number of areas:

Data Rate

- there may be differences of magnitude between the data transfer rates

Application

- the use to which a device is put has an influence on the software

Complexity of Control

- the effect on the operating system is filtered by the complexity of the I/O module that controls the device

Unit of Transfer

- data may be transferred as a stream of bytes or characters or in larger blocks

Data Representation

- different data encoding schemes are used by different devices

Error Conditions

- the nature of errors, the way in which they are reported, their consequences, and the available range of responses differs from one device to another

Data rates

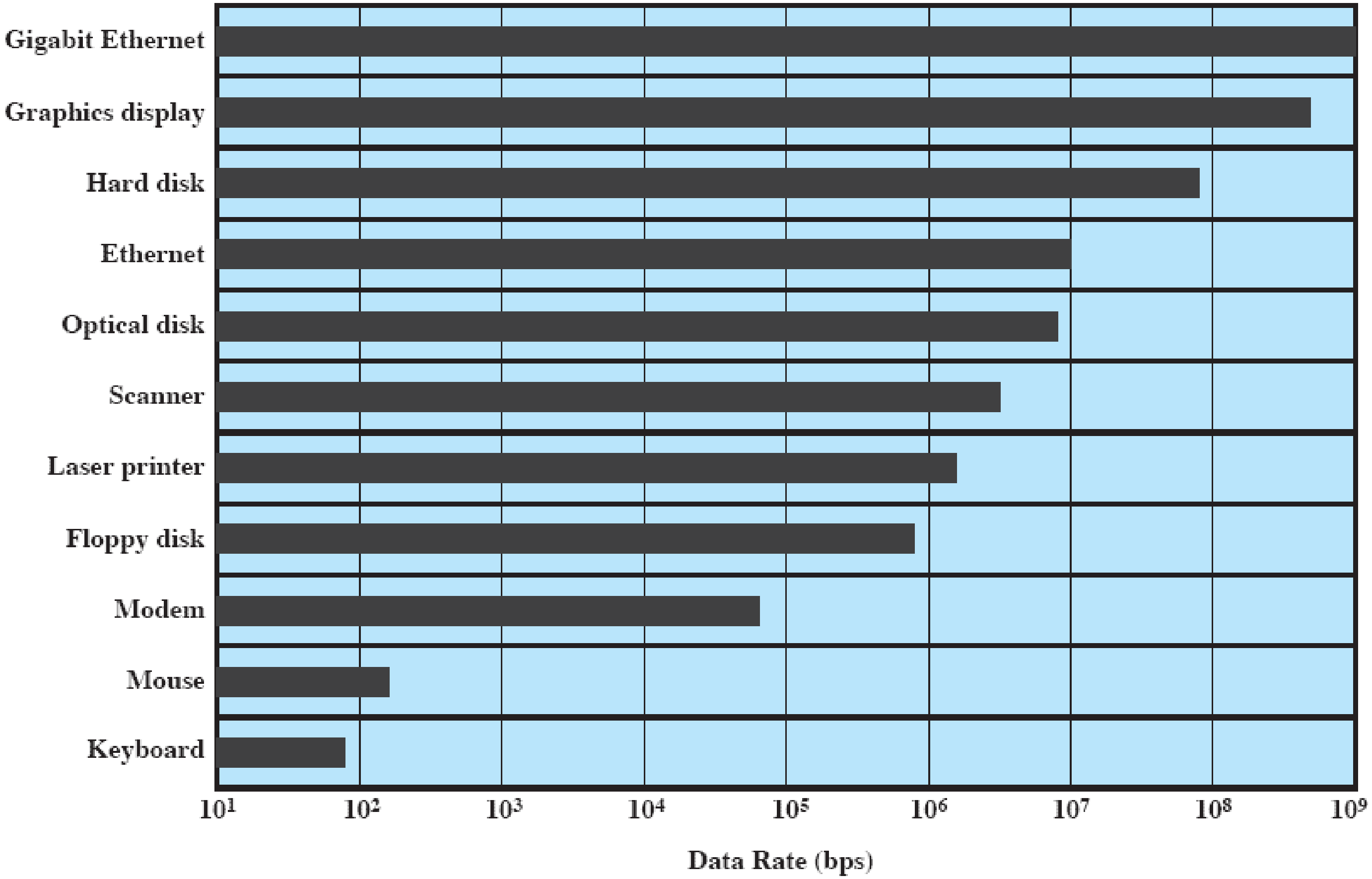


Figure 11.1 Typical I/O Device Data Rates

OS Design objectives for I/O facility

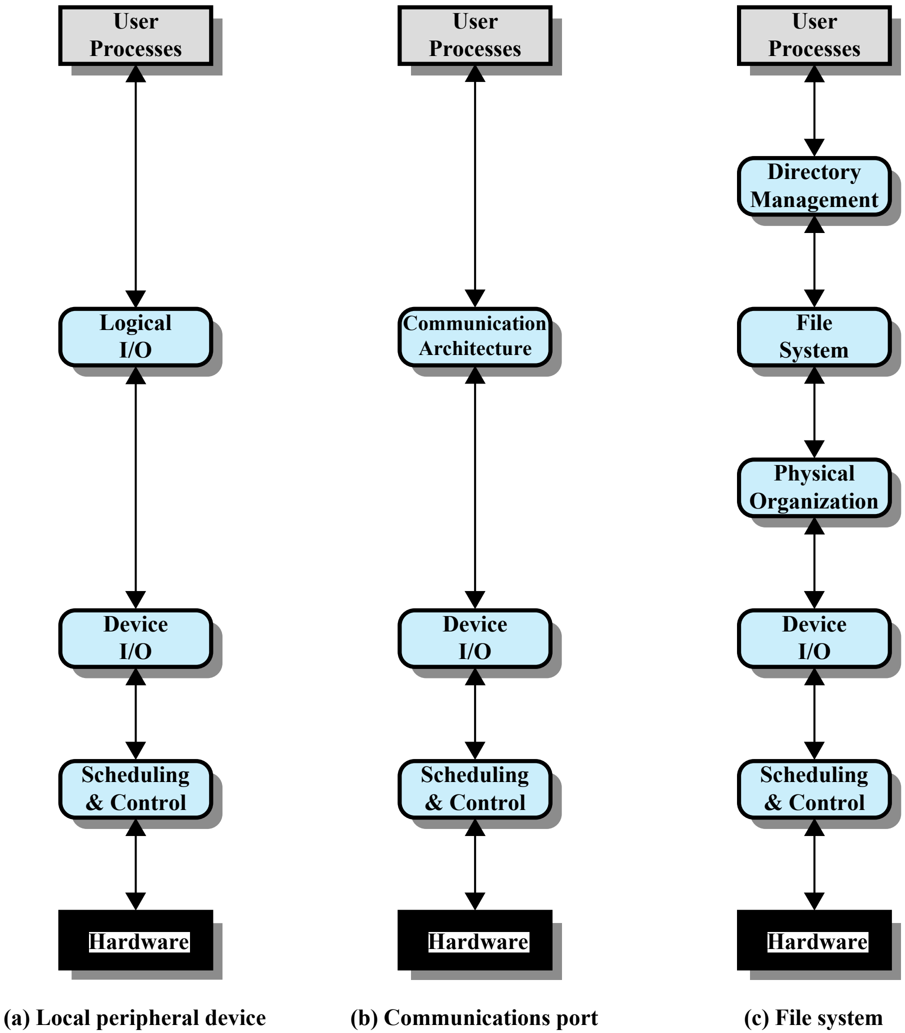
- **Efficiency**
  - Major effort in I/O design
  - Important because I/O operations often form a bottleneck
  - Most I/O devices are extremely slow compared with main memory and the processor
  - To tackle multiprogramming, swapping, virtual memory introduced…but
  - The area that has received the most 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
    - Hides most of the details of device in lower-level routines

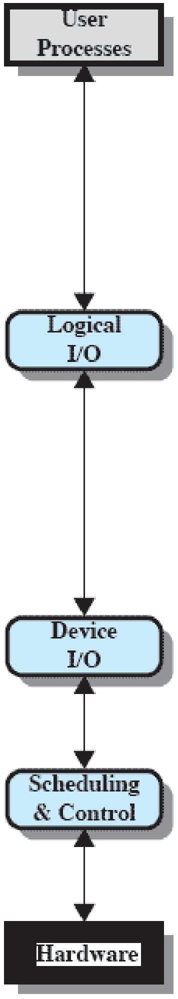
Hierarchical design

- Functions of the operating system should be separated according to their complexity, their characteristic time scale, and their level of abstraction
- Leads to an organization 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

A model of I/O organization



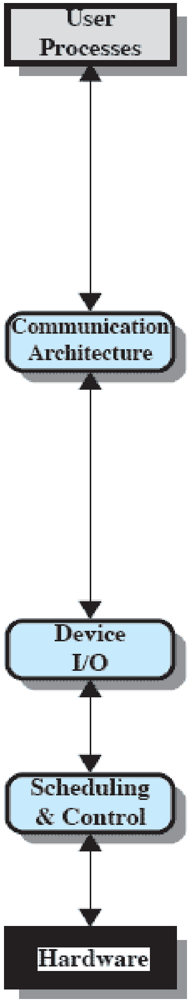
Local peripheral device



(a) Local peripheral device

- Logical I/O :
  - The logical I/O module deals with the device as a logical resource and is not concerned with the details of actually controlling the device.
  - The logical I/O module is concerned with managing general I/O functions on behalf of user processes, allowing them to deal with the device in terms of a device identifier and simple commands such as open, close, read, and write.
- Device I/O :
  - The requested operations and data (buffered characters, records, etc.) are converted into appropriate sequences of I/O instructions, channel commands, and controller orders.
  - Buffering techniques may be used to improve utilization.
- Scheduling and control:
  - The actual queuing and scheduling of I/O operations occurs at this layer, as well as the control of the operations.
  - Interrupts are handled at this layer and I/O status is collected and reported.
  - This is the layer of software that actually interacts with the I/O module and hence the device hardware.

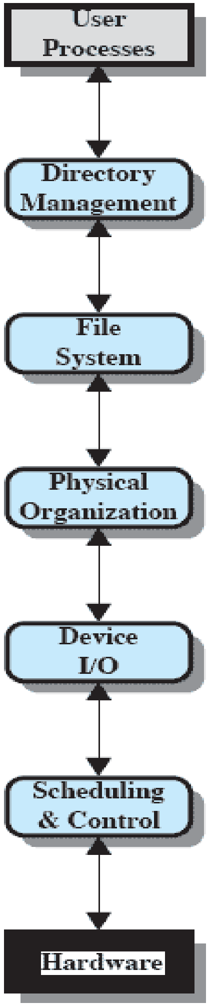
Communication port



(b) Communications port

- The principal difference here is that the logical I/O module is replaced by a communications architecture, which may itself consist of a number of layers. An example is TCP/IP.

File structure



(c) File system

- Directory management:
  - At this layer, symbolic file names are converted to identifiers that either reference the file directly or indirectly through a file descriptor or index table.
  - This layer is also concerned with user operations that affect the directory of files, such as add, delete, and reorganize.

- File system:
  - This layer deals with the logical structure of files and with the operations that can be specified by users, such as open, close, read, and write.
  - Access rights are also managed at this layer.

- Physical organization:
  - Just as virtual memory addresses must be converted into physical main memory addresses, taking into account the segmentation and paging structure, logical references to files and records must be converted to physical secondary storage addresses, taking into account the physical track and sector structure of the secondary storage device.
  - Allocation of secondary storage space and main storage buffers is generally treated at this layer as well.

Organisation of the I/O function

- Three 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 module
    - The processor sends a request for the transfer of a block of data to the DMA module and is interrupted only after the entire block has been transferred.

Techniques for performing I/O

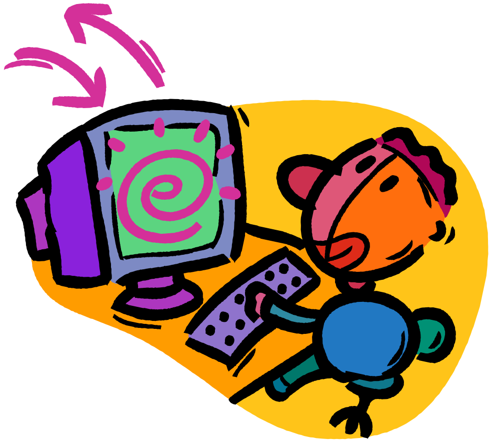
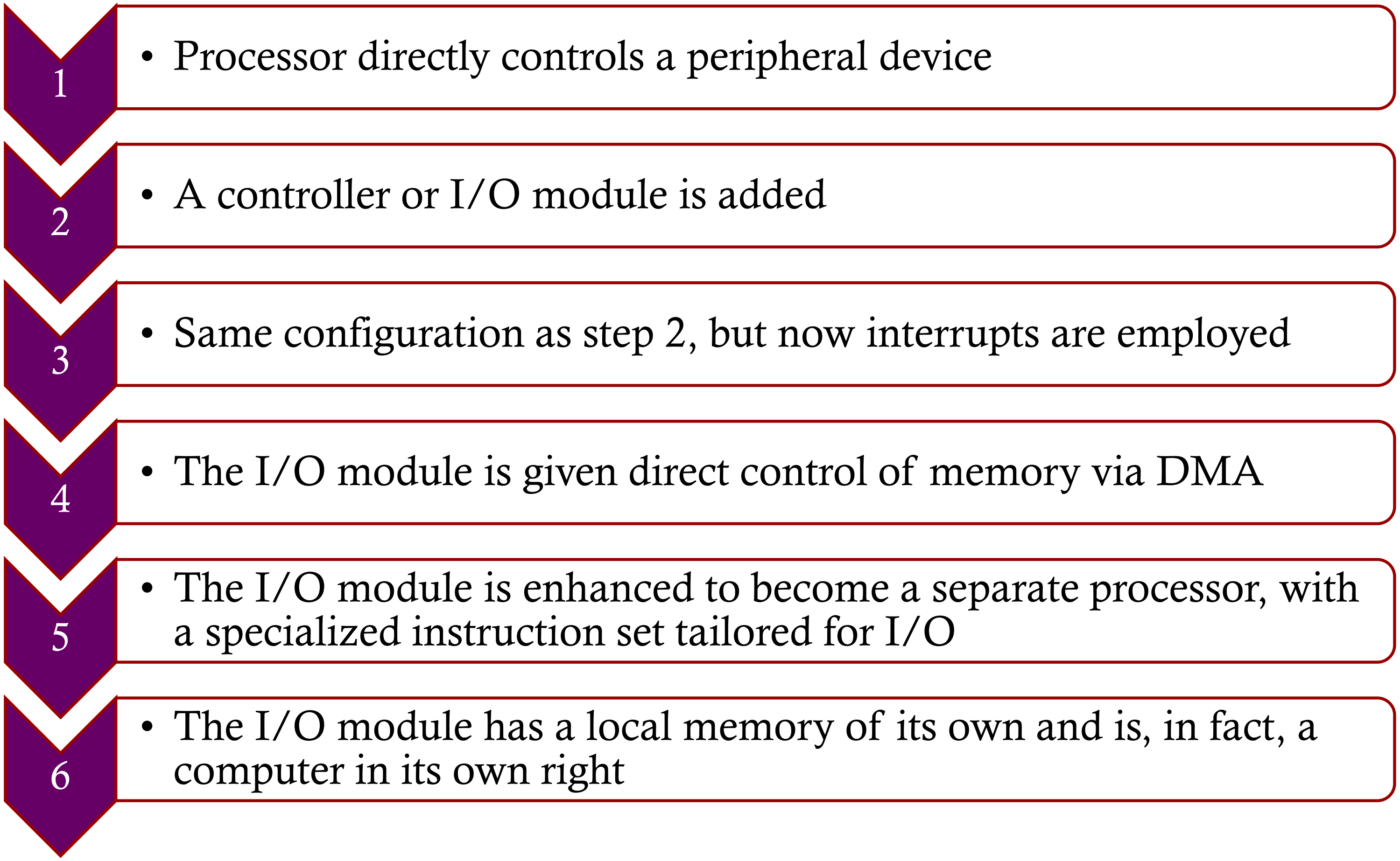


Table 11.1 I/O Techniques

	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)

- In most computer systems, DMA is the dominant form of transfer that must be supported by the operating system.

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 specialized 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

- The DMA unit is capable of mimicking the processor and of taking over control of the system bus just like a processor.
  - It needs to do this to transfer data to/from memory over the system bus.
- When the processor wishes to read or write a block of data, it issues a command to the DMA module by sending to the DMA module the following information:
  - Whether a read or write is requested, using the read or write control line between the processor and the DMA module
  - The address of the I/O device involved, communicated on the data lines
  - The starting location in memory to read from or write to, communicated on the data lines and stored by the DMA module in its address register
  - The number of words to be read or written, again communicated via the data lines and stored in the data count register
- The processor then continues with other work.
  - It has delegated this I/O operation to the DMA module. The DMA module transfers the entire block of data, one word at a time, directly to or from memory, without going through the processor.
  - When the transfer is complete, the DMA module sends an interrupt signal to the processor.
- The processor is involved only at the beginning and end of the transfer

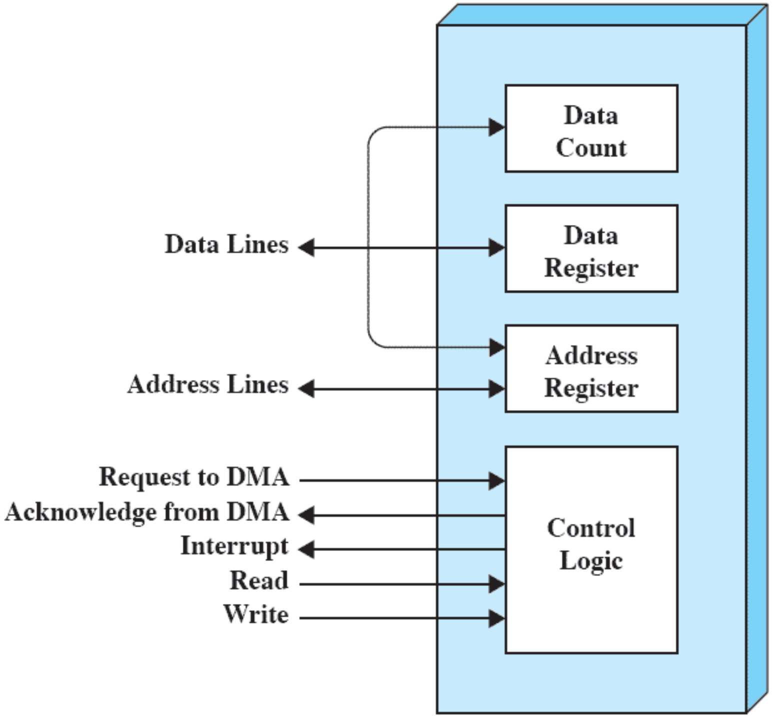
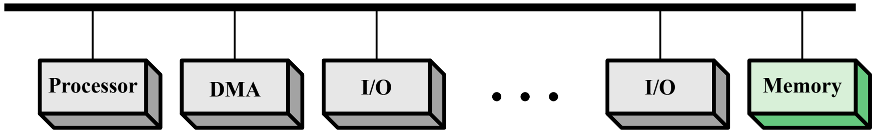
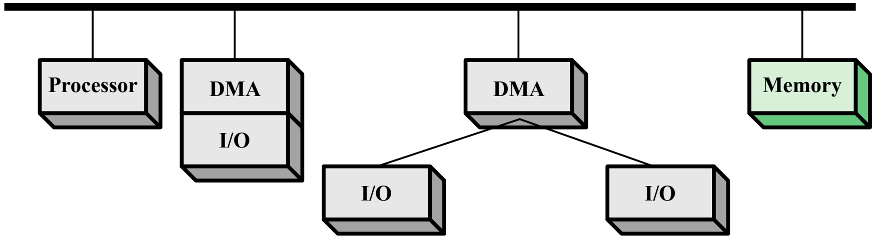


Figure 11.2 Typical DMA Block Diagram

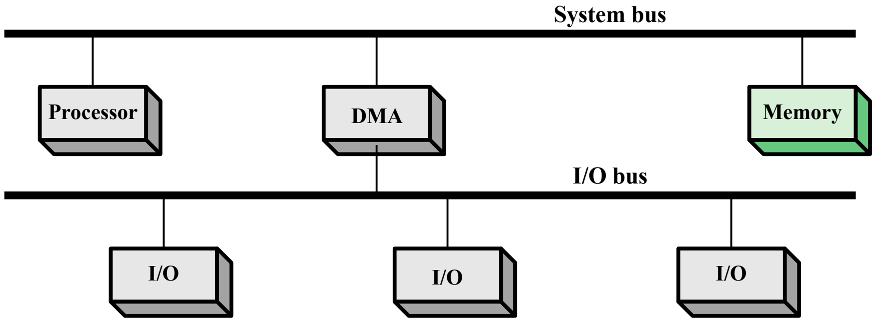
DMA Configurations



(a) Single-bus, detached DMA



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



(c) I/O bus

I/O - Buffering

- Consider an I/O command to transfer 512 bytes from disk to virtual address 1000 to 1511 of the process

```
Read_Block[1000, disk]
```

- I/O and process swapping may interact, and possibly cause deadlock, due to the time delayed nature of I/O operations.

- It can be convenient to transfer input data in advance of an Input request, and to transfer output data some time after an Output request is made.

- Use buffering.

Buffering

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

Block-oriented device

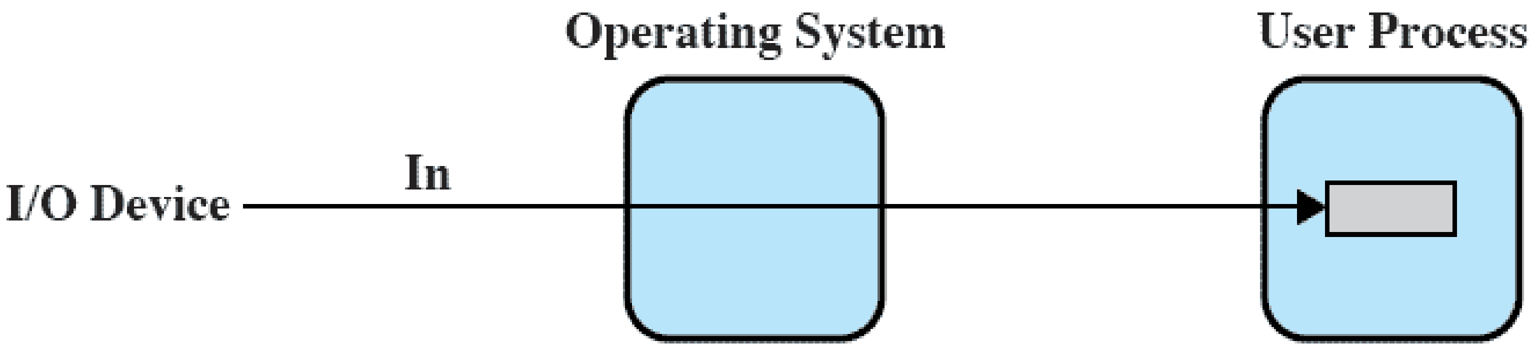
- stores information in blocks that are usually of fixed size
- transfers are made one block at a time
- possible to reference data by its block number
- disks and USB keys are examples

Stream-oriented device

- 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 needs



(a) No buffering

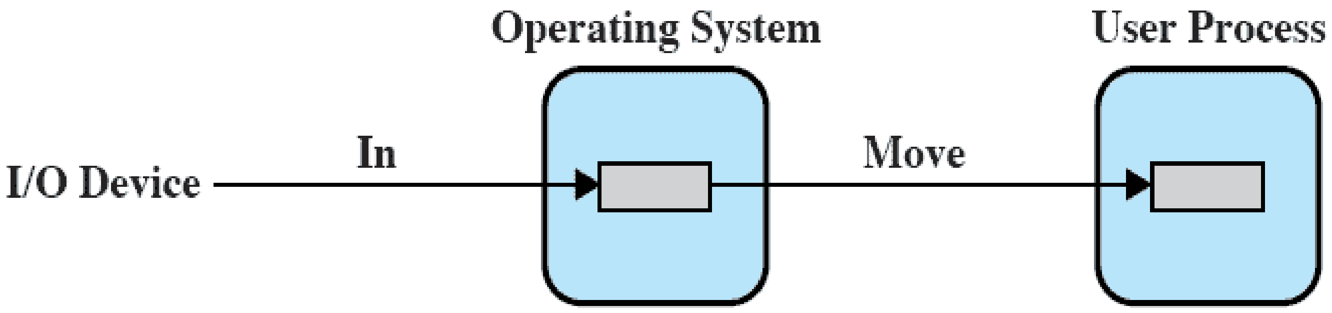
Suppose, **T**: time required to input one bock

C: is the computation time between input requests

T+C: execution time per block

Single buffer

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



(b) Single buffering

Suppose, **T**: time required to input one bock

C: is the computation time between input requests

max[T,C] + M: execution time per block

M: time to move data from system buffer to user memory

Block-oriented single buffer

- Input transfers are made to the system buffer

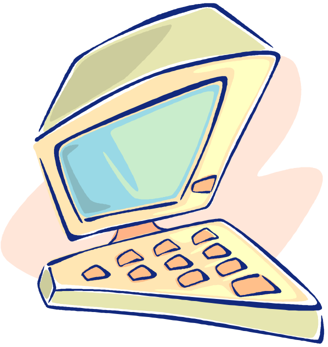
- Reading ahead/anticipated input
  - is done in the expectation that the block will eventually be needed
  - when the transfer is complete, the process moves the block into user space and immediately requests another block

- Generally provides a speedup compared to the lack of system buffering

- Disadvantages:
  - complicates the logic in the operating system (swap vs buffer disc)
  - 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 signaling the end of a line
  - output to the terminal is similarly one line at a time

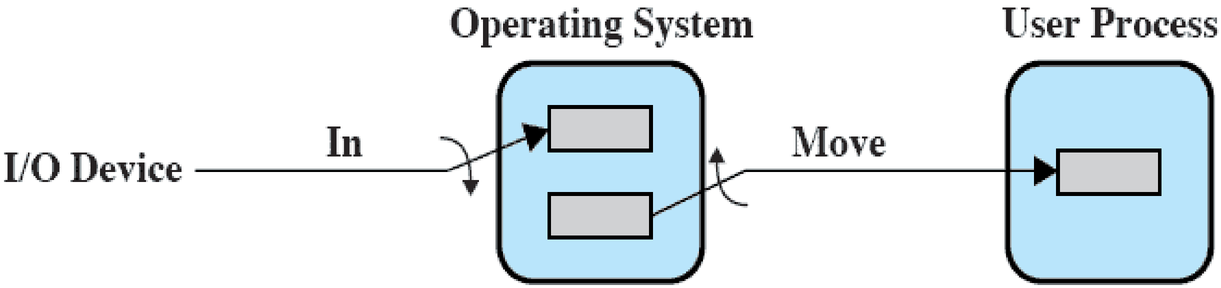


- Byte-at-a-time operation
  - used on forms-mode terminals
  - when each keystroke is significant
  - other peripherals such as sensors and controllers

09/10/2018

Double buffering

- 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

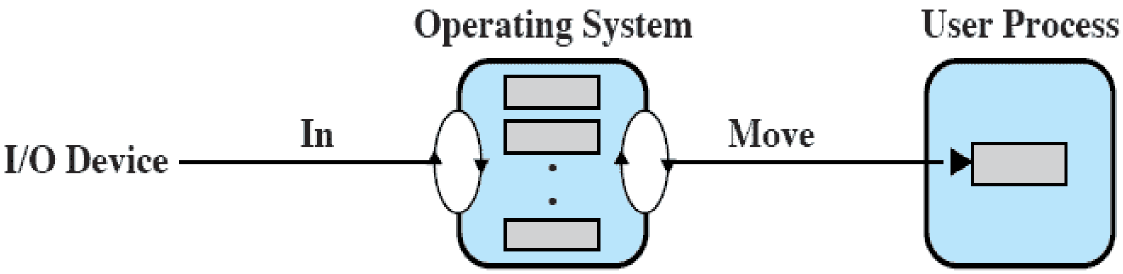
Suppose, **T**: time required to input one bock

C: is the computation time between input requests

max[T,C]: execution time per block

Circular buffer

- Double buffering may be inadequate if the process performs rapid bursts of I/O
- More that two buffers are used
- Each individual buffer is one unit in a circular buffer
- Used when I/O operation must keep up with process
- Bounded-buffer producer/consumer model

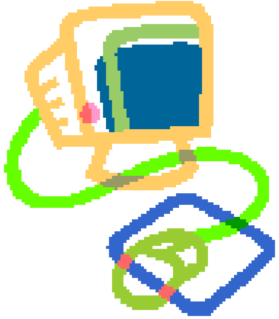


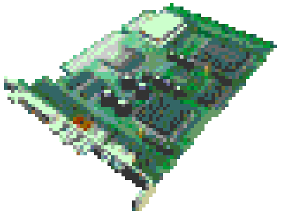
(d) Circular buffering

Utility of buffering

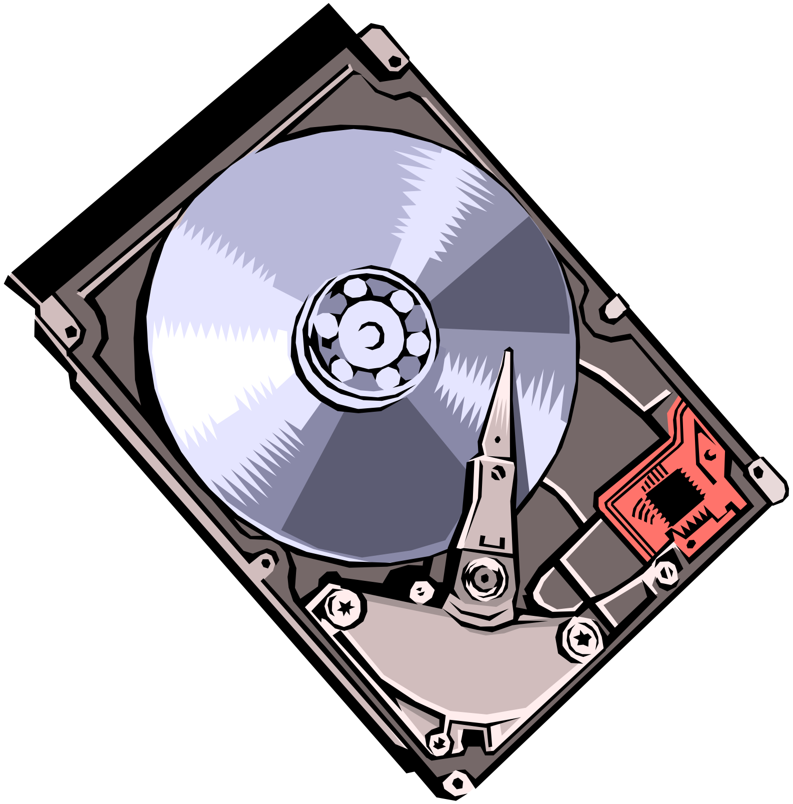
- Technique that smooths out peaks in I/O demand
  - with enough demand eventually all buffers become full and their advantage is lost

- 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





Disk Scheduling

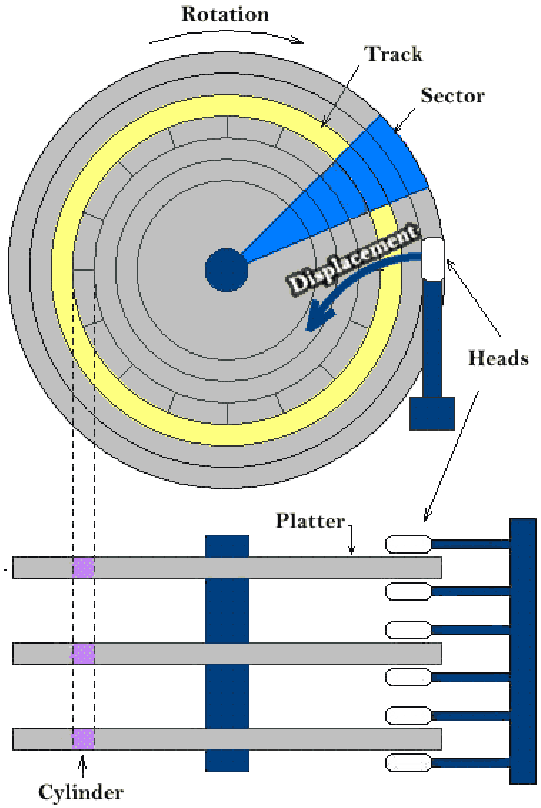


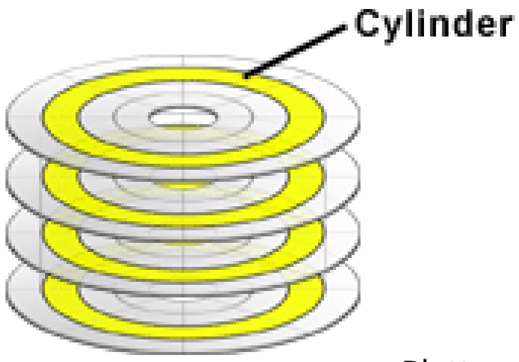
Disk I/O

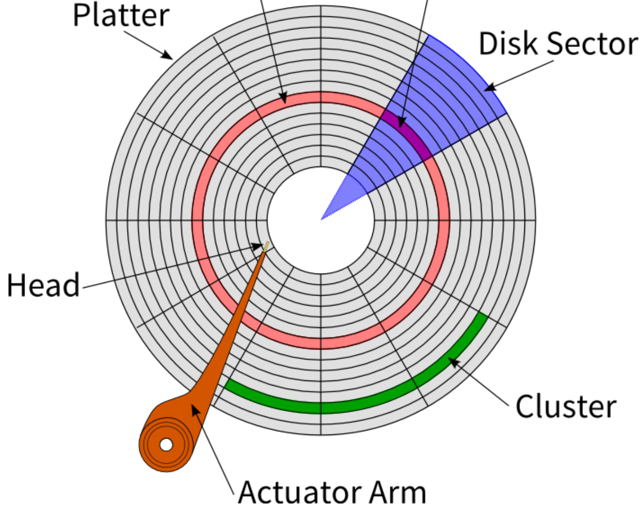
- Disk I/O has the greatest impact on overall system performance

- Two of the most widely used approaches are:
  - Disk scheduling
  - Disk cache

Disk component







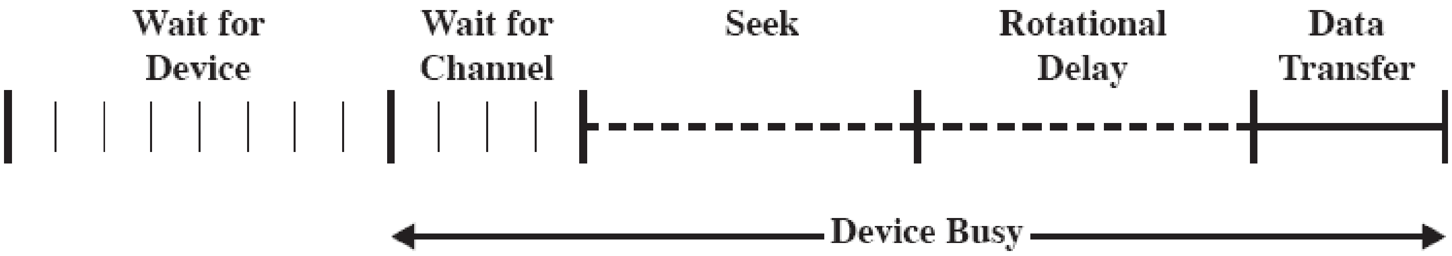
Disk performance parameters

- 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

Positioning the Read/Write heads

- 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**

Disk performance parameters



Disk scheduling policies

- The dominant component restricting disk drive performance is the **seek time.**

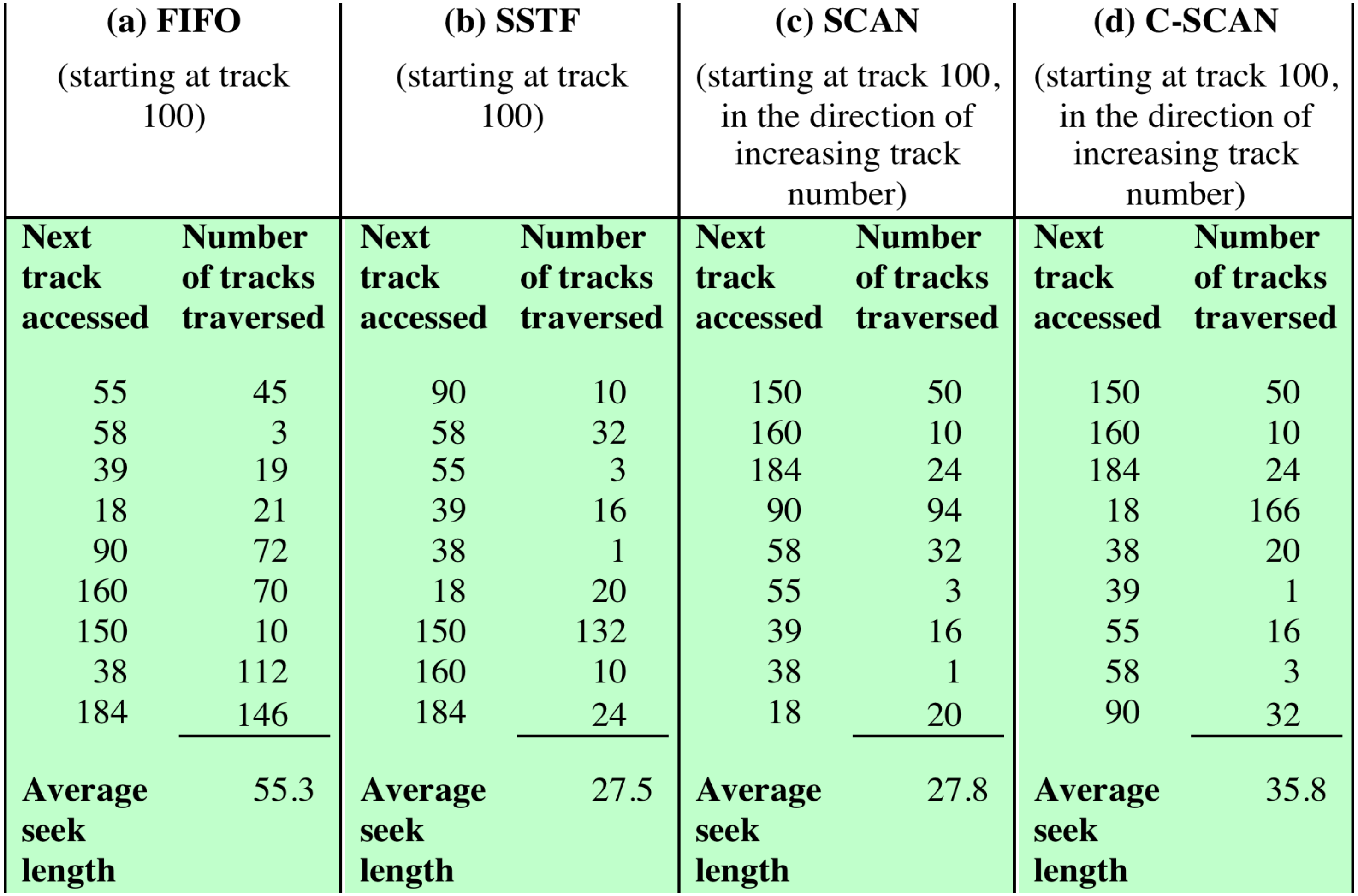
- If sector accesses are randomly distributed over many tracks, disk I/O performance will be poor.

- We must reduce the impact of the seek time component of disk access time.

- In the same way that the *optimal page-replacement strategy* is useful for providing a base-line metric for memory system performance, the *random scheduling strategy* provide a benchmark against which to evaluate disk scheduling techniques.

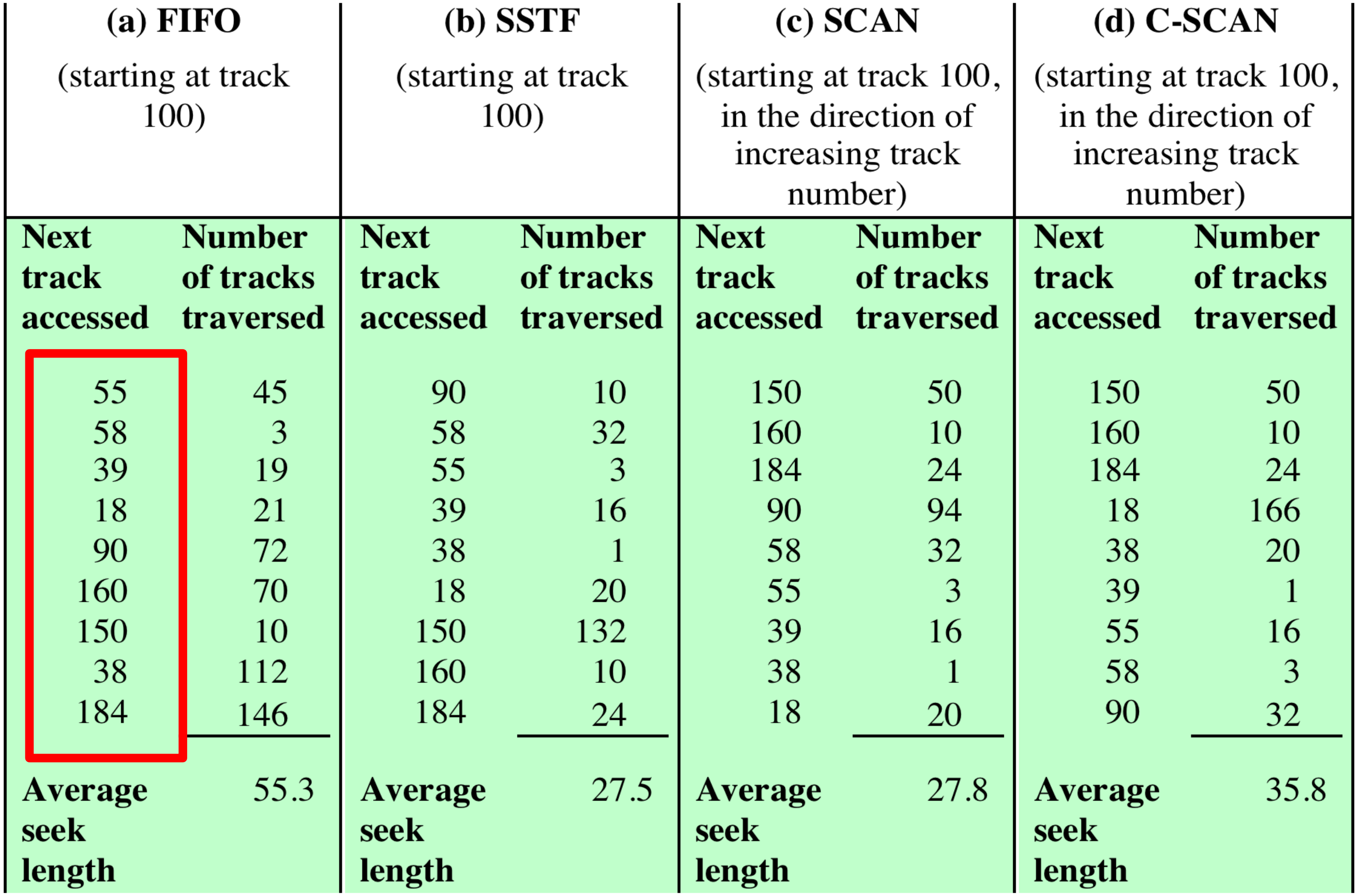
- **Random Scheduling**
  - queue disk I/O requests in random order
  - tracks will be visited at random
  - this gives the *worst possible* performance

Disk scheduling algorithms



(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

Disk scheduling algorithms

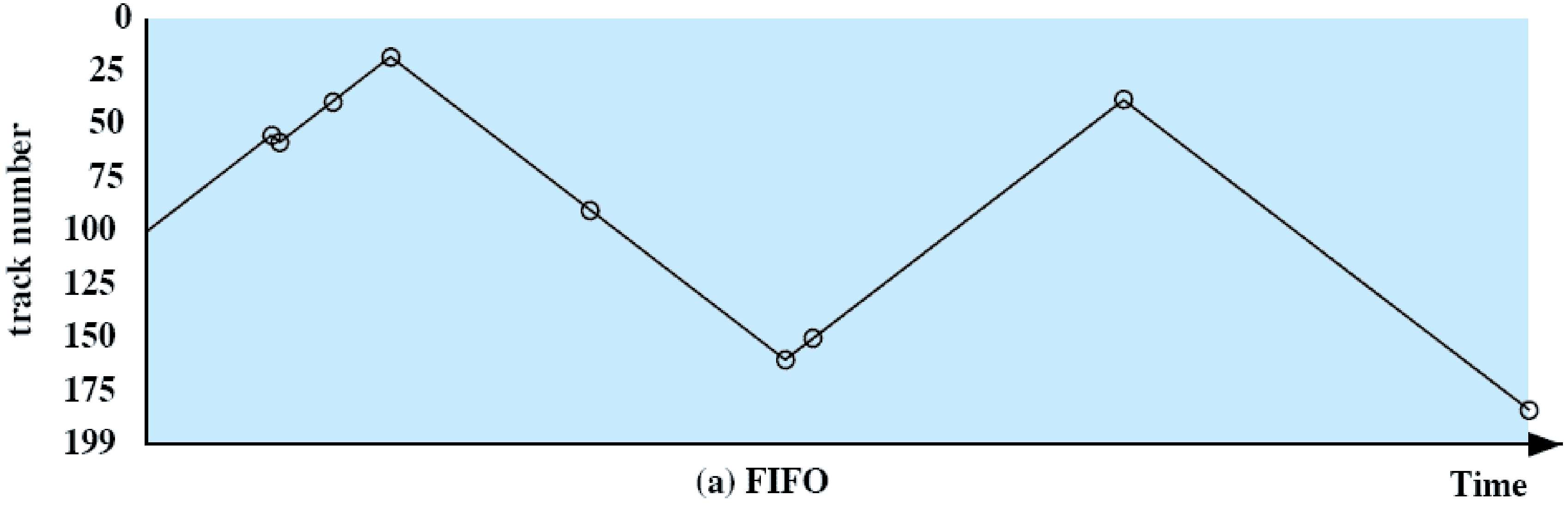


(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

First-In, First-Out (FIFO)

55, 58, 39, 18, 90, 160, 150, 38, 184

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

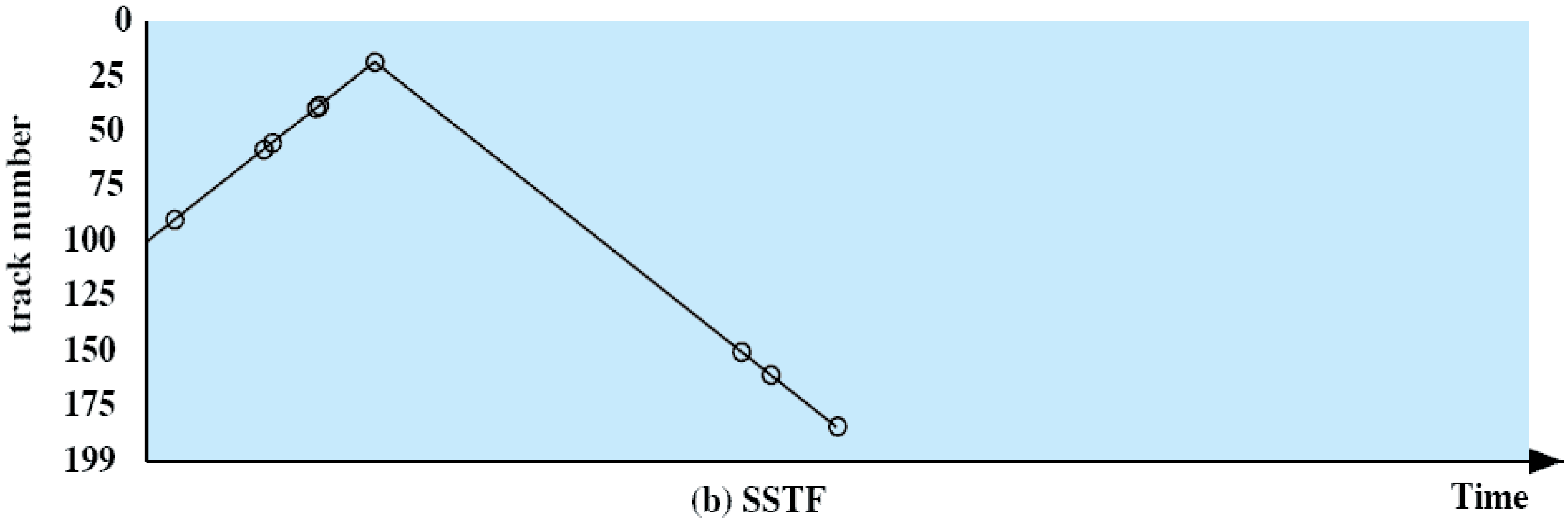


(a) FIFO

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

55, 58, 39, 18, 90, 160, 150, 38, 184

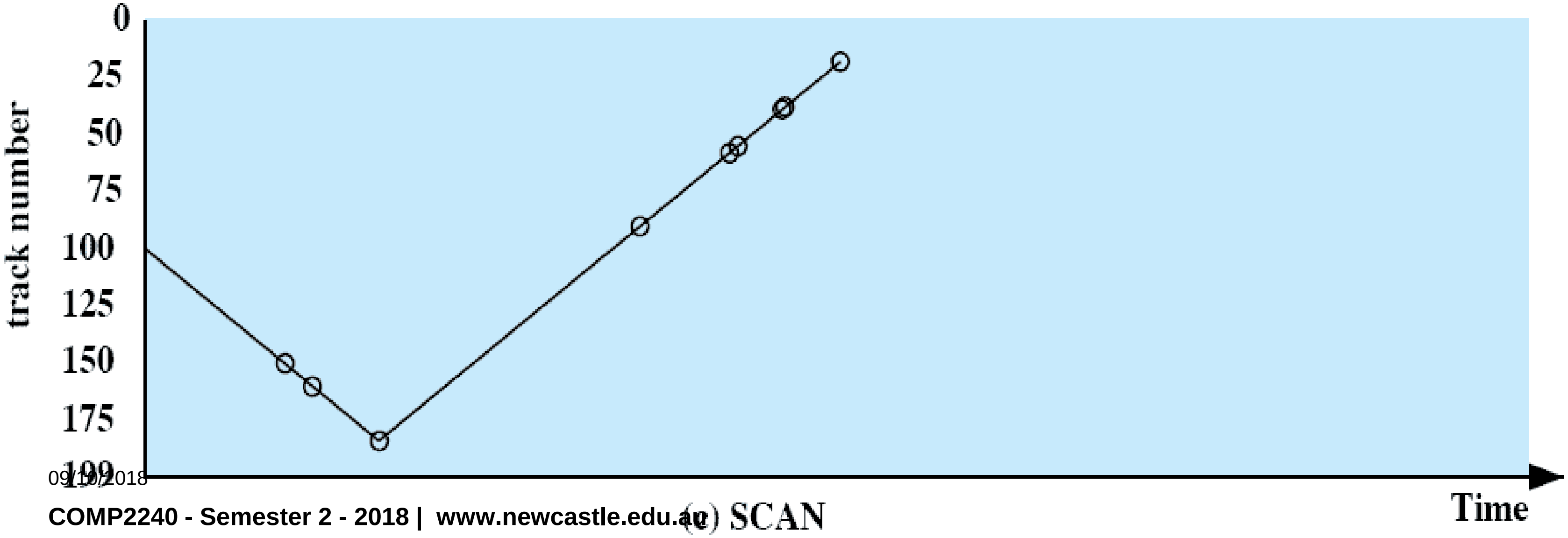


(b) SSTF

SCAN/LOOK

55, 58, 39, 18, 90, 160, 150, 38, 184

- 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

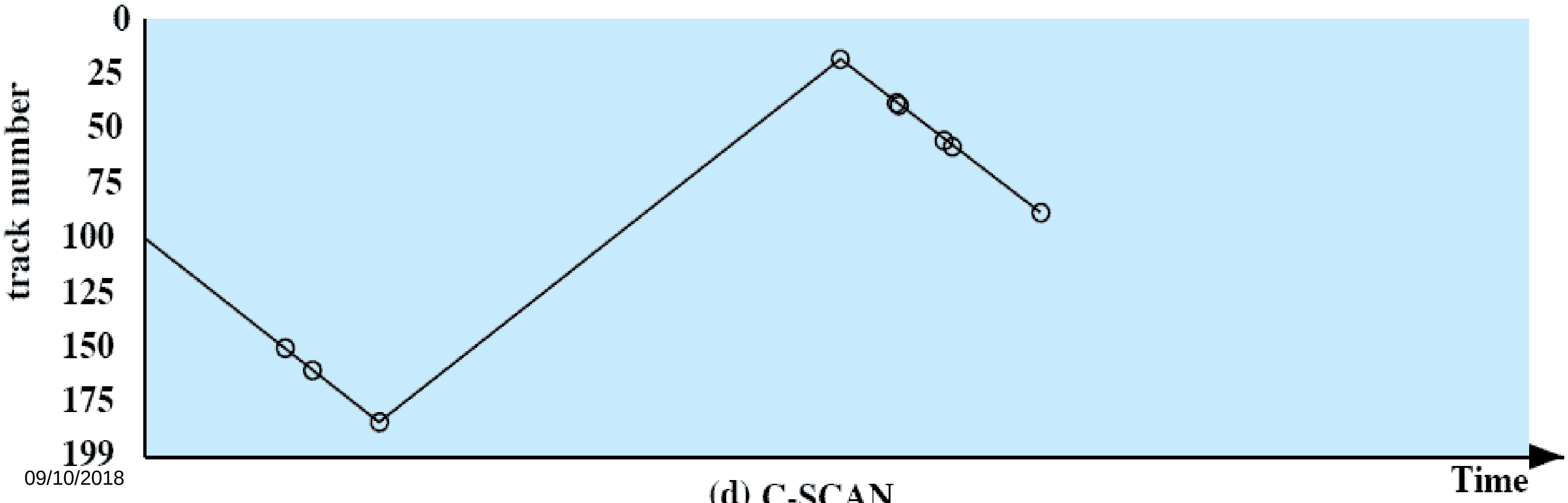


(c) SCAN

C-SCAN/C-LOOK

55, 58, 39, 18, 90, 160, 150, 38, 184

- 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

Issues

- With SSTF, SCAN, and C-SCAN, it is possible that the arm may not move for a considerable period of time.
  - For example, if one or a few processes have high access rates to one track, they can monopolize the entire device by repeated requests to that track.
  - High-density multi-surface disks are more likely to be affected by this characteristic than lower-density disks and/or disks with only one or two surfaces.

- To avoid this “arm stickiness,” the disk request queue can be segmented, with one segment at a time being processed completely.

- Two examples of this approach are N-Step-SCAN and FSCAN.

N-Step-SCAN

- Segments the disk request queue into sub-queues of length *N*
- Sub-queues are processed one at a time, using SCAN
- While a queue is being processed new requests must be added to some other queue
- If fewer than *N* requests are available at the end of a scan, all of them are processed with the next scan
- With large values of *N , the performance of N-Step-SCAN approaches* that of SCAN
  - with a value of *N = 1 , the FIFO policy is adopted*

Priority

- Imposed outside of disk management to meet other objectives within OS
- Not intended to optimise disk use
- Example: interactive (and possibly short batch) jobs are given higher priority than longer batch or high computation jobs.
  - This flushes lots of short jobs through the system quickly -> good interactive response time. But long jobs are slow.

RAID

RAID

- With the use of multiple disks, there is a wide variety of ways in which the data can be organized and in which redundancy can be added to improve reliability

- Redundant Array of Independent Disks

- Consists of seven levels, zero through six
  - These levels do not imply a hierarchical relationship but designate different design architecture

RAID Characteristics

RAID design architectures share three characteristics:

1. RAID is a set of physical disk drives viewed by the operating system as a single logical drive
2. Data are distributed across the physical drives of an array in a scheme known as striping
3. Redundant disk capacity is used to store parity information, which guarantees data recoverability in case of a disk failure
   - Not supported by RAID 0 and RAID 1

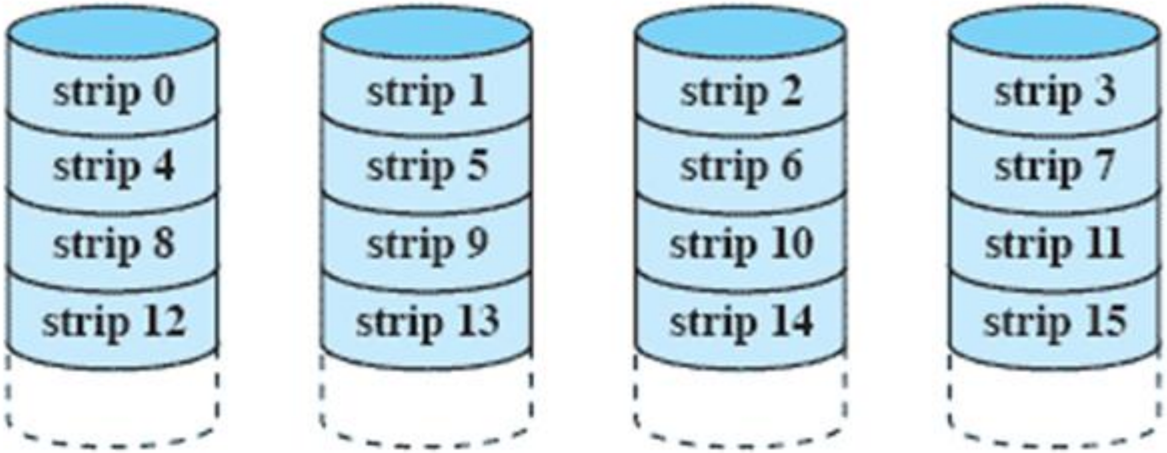
RAID Levels

Category	Level	Description	Disks required	Data availability	Large I/O data transfer capacity	Small I/O request rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

N = number of data disks; m proportional to log N

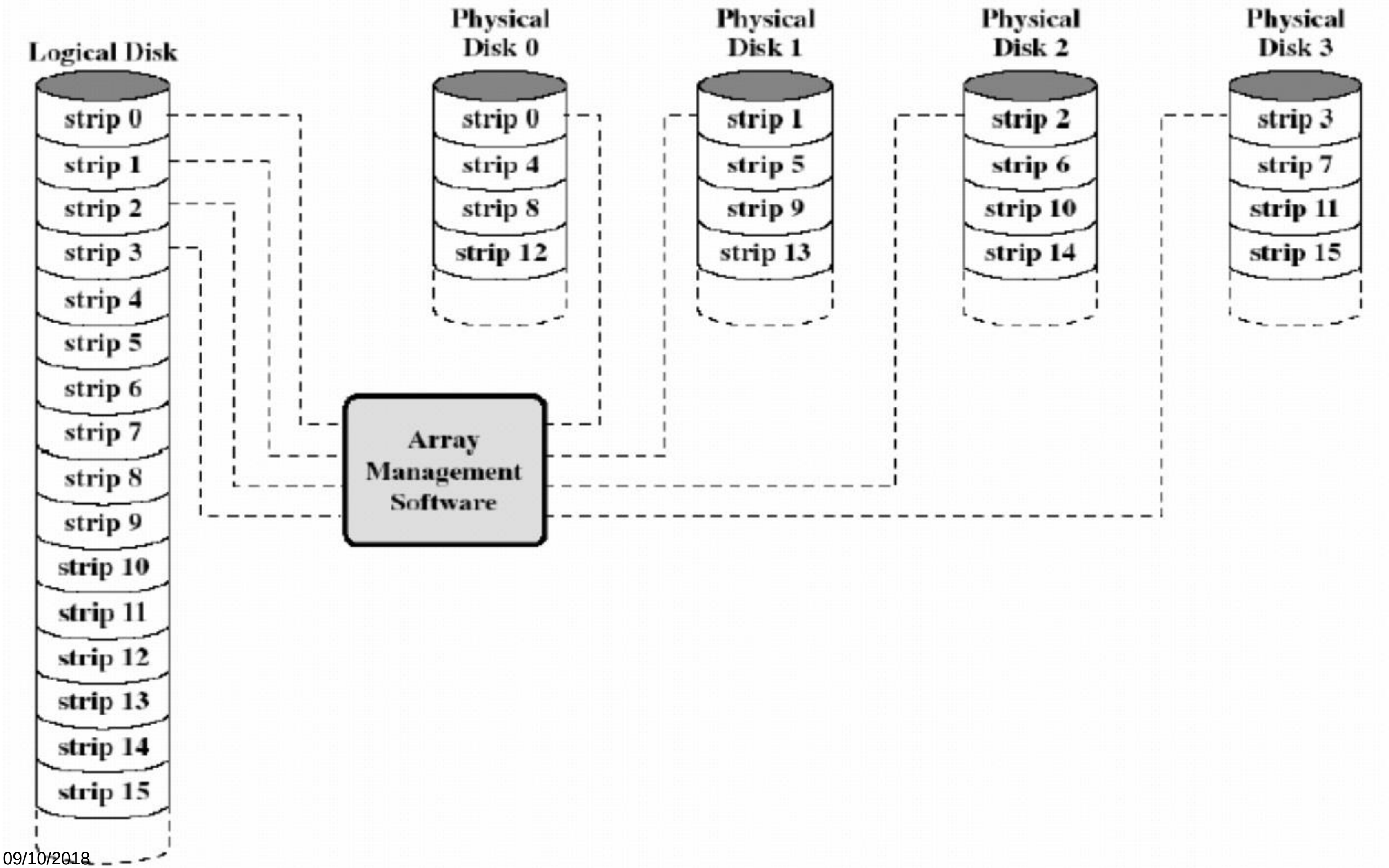
RAID Level 0

- 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)

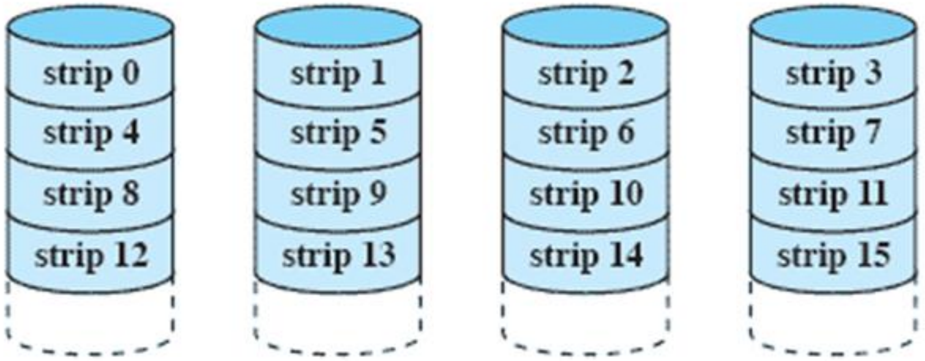
Disk strips


Logical Disk
strip 0
strip 1
strip 2
strip 3
strip 4
strip 5
strip 6
strip 7
strip 8
strip 9
strip 10
strip 11
strip 12
strip 13
strip 14
strip 15
Physical Disk 0
strip 0
strip 4
strip 8
strip 12
Physical Disk 1
strip 1
strip 5
strip 9
strip 13
Physical Disk 2
strip 2
strip 6
strip 10
strip 14
Physical Disk 3
strip 3
strip 7
strip 11
strip 15
Array Management Software

RAID Level 0

- **In an *n -disk array, the first n logical strips are physically stored as*** the first strip on each of the *n disks, forming the first* **stripe***; the second n strips* are distributed as the second strips on each disk; and so on.

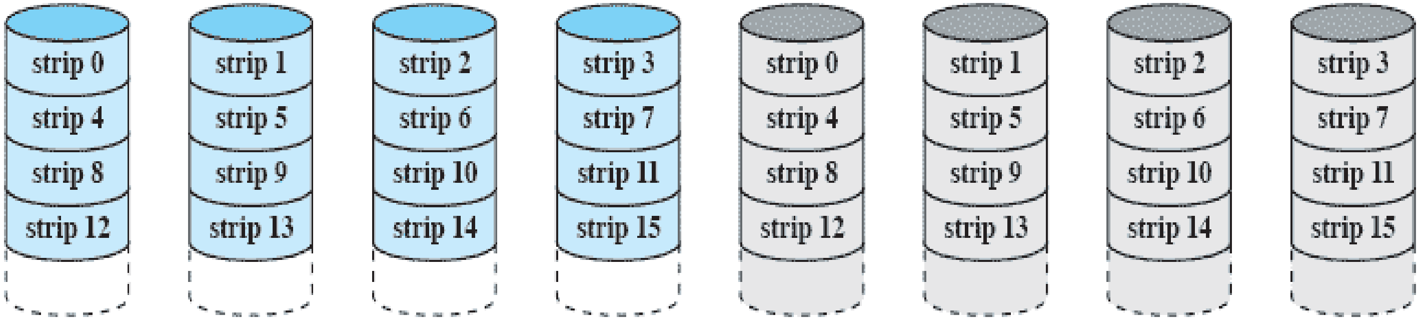
- The advantage of this layout is that if a single I/O request consists of multiple logically contiguous strips, then up to *n strips for that request can be handled in parallel, greatly reducing the* I/O transfer time.



(a) RAID 0 (non-redundant)

RAID Level 1

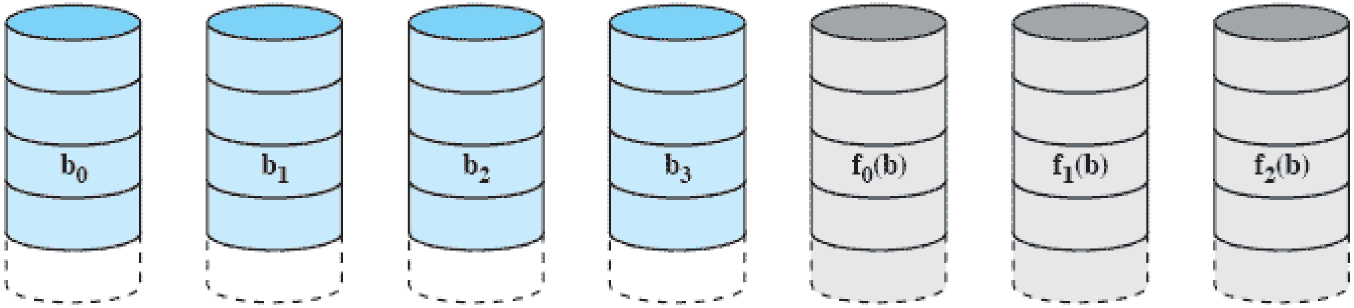
- Redundancy is achieved by the simple expedient of duplicating all the data
- Advantages:
  - Read request can be served by either of the two disks containing the data
  - Write needs both strips to be updated but there is no “write penalty”
  - When a drive fails the data may still be accessed from the second drive
- Principal disadvantage is the cost



(b) RAID 1 (mirrored)

RAID Level 2

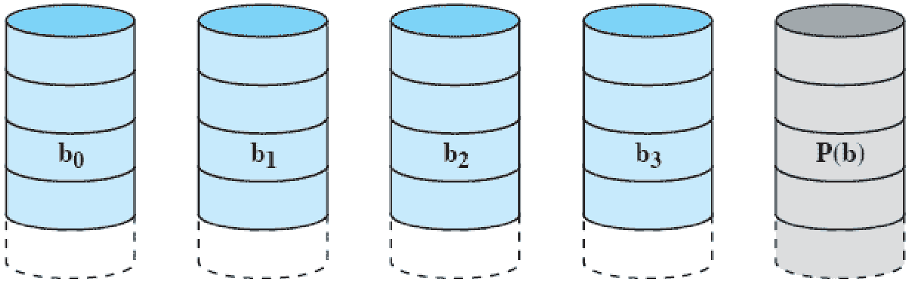
- Makes use of a **parallel access technique**
  - All member disks participate in the execution of every I/O request
- Data striping is used
  - Strips are very small
- Error-correcting code is calculated across corresponding bits on each data disk, and the bits of the code are stored in the corresponding bit positions on multiple parity disks
  - Typically a Hamming code is used - correct single-bit errors and detect double-bit errors
- Effective choice in an environment in which many disk errors occur



(c) RAID 2 (redundancy through Hamming code)

RAID Level 3

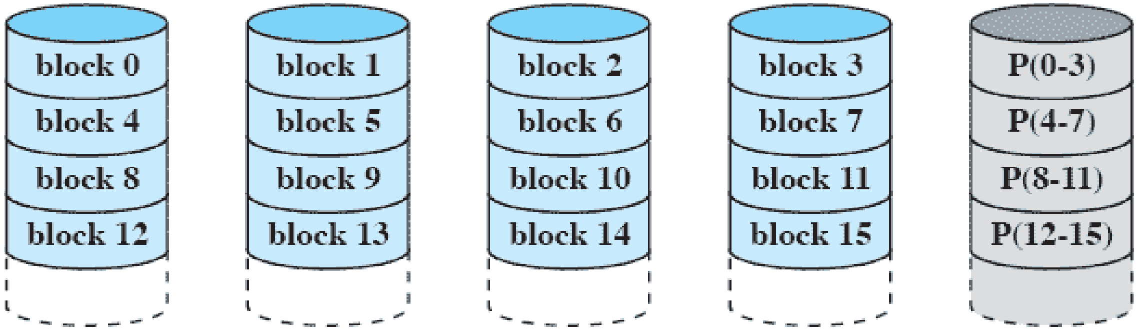
- Requires only a single redundant disk, no matter how large the disk array
  - a simple parity bit is computed for the set of individual bits in the same position on all of the data disks
- Employs **parallel access technique**
  - With data distributed in small strips
- Can achieve very high data transfer rates



(d) RAID 3 (bit-interleaved parity)

RAID Level 4

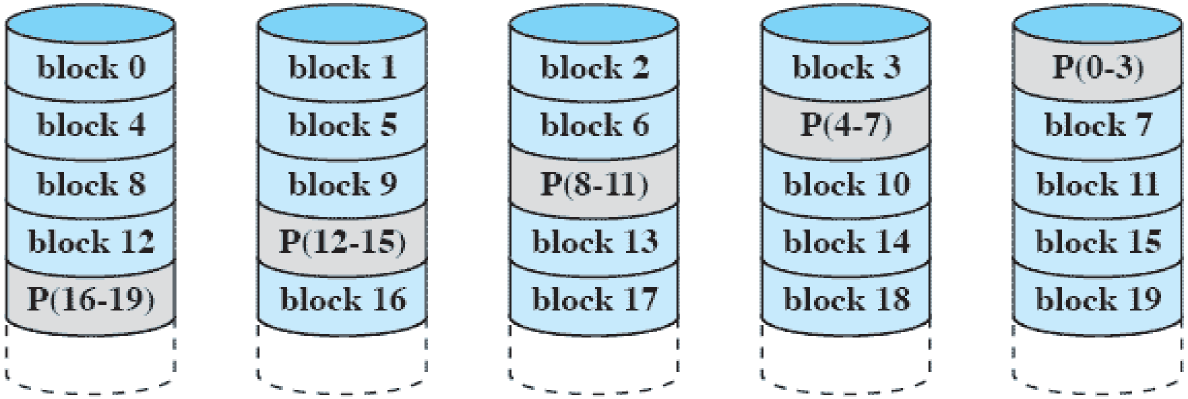
- Makes use of an **independent access technique**
  - Each member disk operates independently, so that separate I/O requests can be satisfied in parallel
- A bit-by-bit parity strip is calculated across corresponding strips on each data disk, and the parity bits are stored in the corresponding strip 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

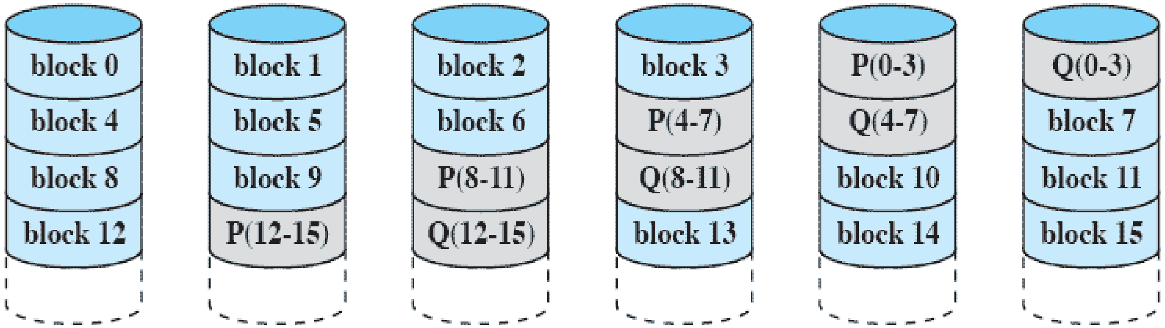
- Similar to RAID-4 but distributes the parity bits across all disks
- Typical allocation is a round-robin scheme
- Avoid the potential I/O bottle-neck of the single parity disk in RAID 4
- Has the characteristic that the loss of any one disk does not result in data loss



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

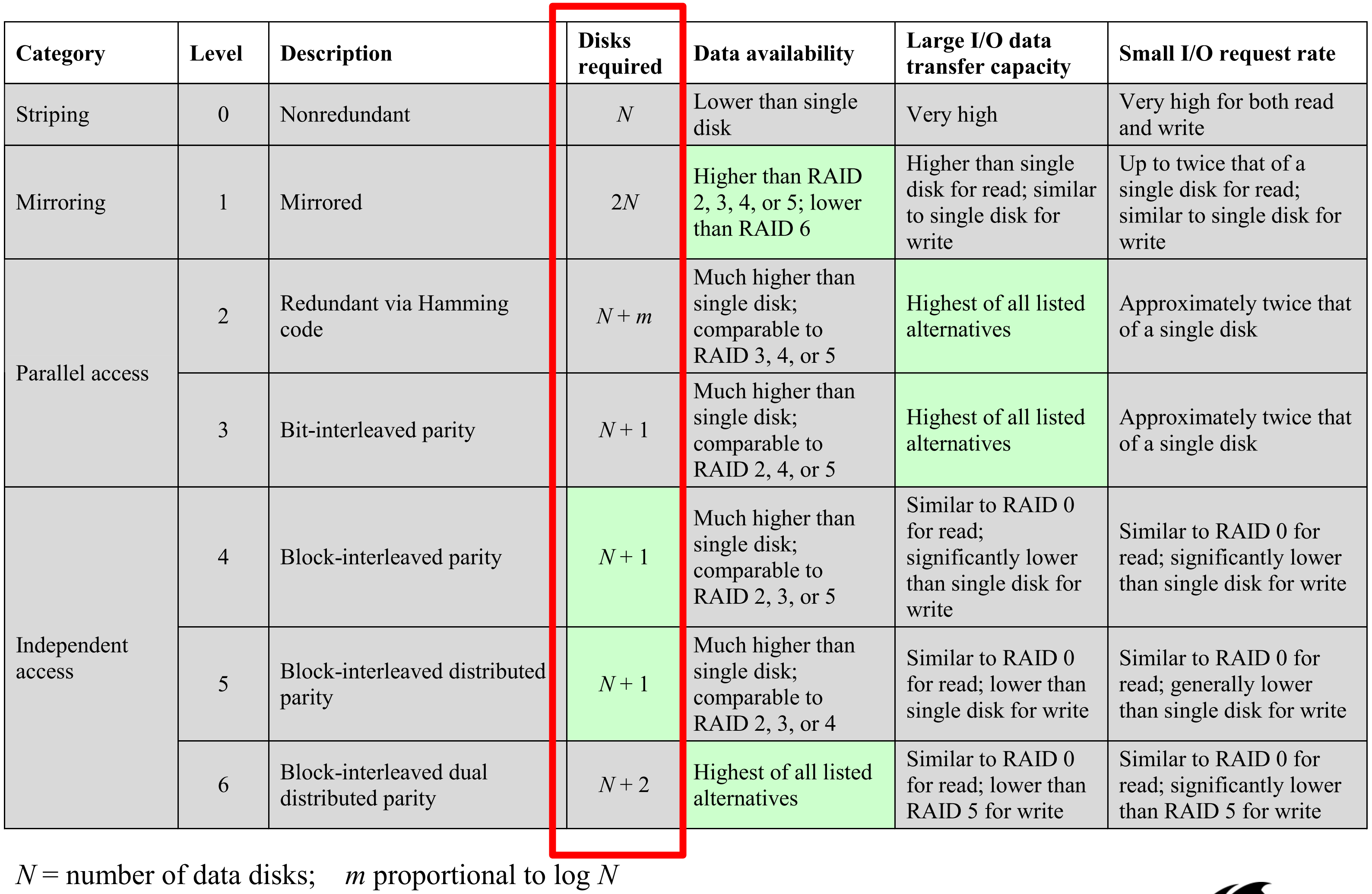
RAID Level 6

- Two different parity calculations are carried out and stored in separate blocks on different disks
- Provides extremely high data availability
- Incurs a substantial write penalty because each write affects two parity blocks



(g) RAID 6 (dual redundancy)

RAID Levels



Category	Level	Description	Disks required	Data availability	Large I/O data transfer capacity	Small I/O request rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

N = number of data disks; m proportional to log N

Summary

- I/O function is generally broken up into a number of layers
  - Lower layers deal with detains and closer to physical functions
  - Higher layers deal with I/O in a logical and generic fashion
  - Changes in HW parameter need not to affect most of the I/O SW
- A key aspect of I/O is the use of buffers that are controlled by I/O utilities rather than by application processes
  - Buffering smooths out the differences between the internal speed of the computer system and the speed of I/O devices
  - Decouples the actual I/O transfer from the address space of the application process
  - Allows OS more flexibility in memory management

Summary

- Performance of disk storage subsystem is of vital concern
- Disk scheduling is to satisfy the I/O requests on the disk in a way to minimize the seek time of the disk
  - FCFS, STTF, SCAN, C-SCAN are some algorithms used
- RAID is an architecture to organize data over multiple disks to improve data transfer capacity and data reliability
  - RAID scheme consists of seven level zero through six
- .

References

- **Operating Systems – Internal and Design Principles**
  - By William Stallings
- Chapter 11