Unit	Details		Lectures
I	**Operating System Overview:** Objectives and Functions, Evolution, Achievements, Modern Operating Systems, Fault tolerance, OS design considerations for multiprocessor and multicore, overview of different operating systems **Processes:** Process Description and Control.		12
II	**Threads, Concurrency:** Mutual Exclusion and Synchronization.		12
III	**Concurrency:** Deadlock and Starvation, **Memory:** Memory Management, Virtual Memory.		12
IV	**Scheduling:** Uniprocessor Scheduling, Multiprocessor and Real-Time Scheduling		12
V	**IO and File Management:** I/O Management and Disk Scheduling, File Management, **Operating System Security.**		12

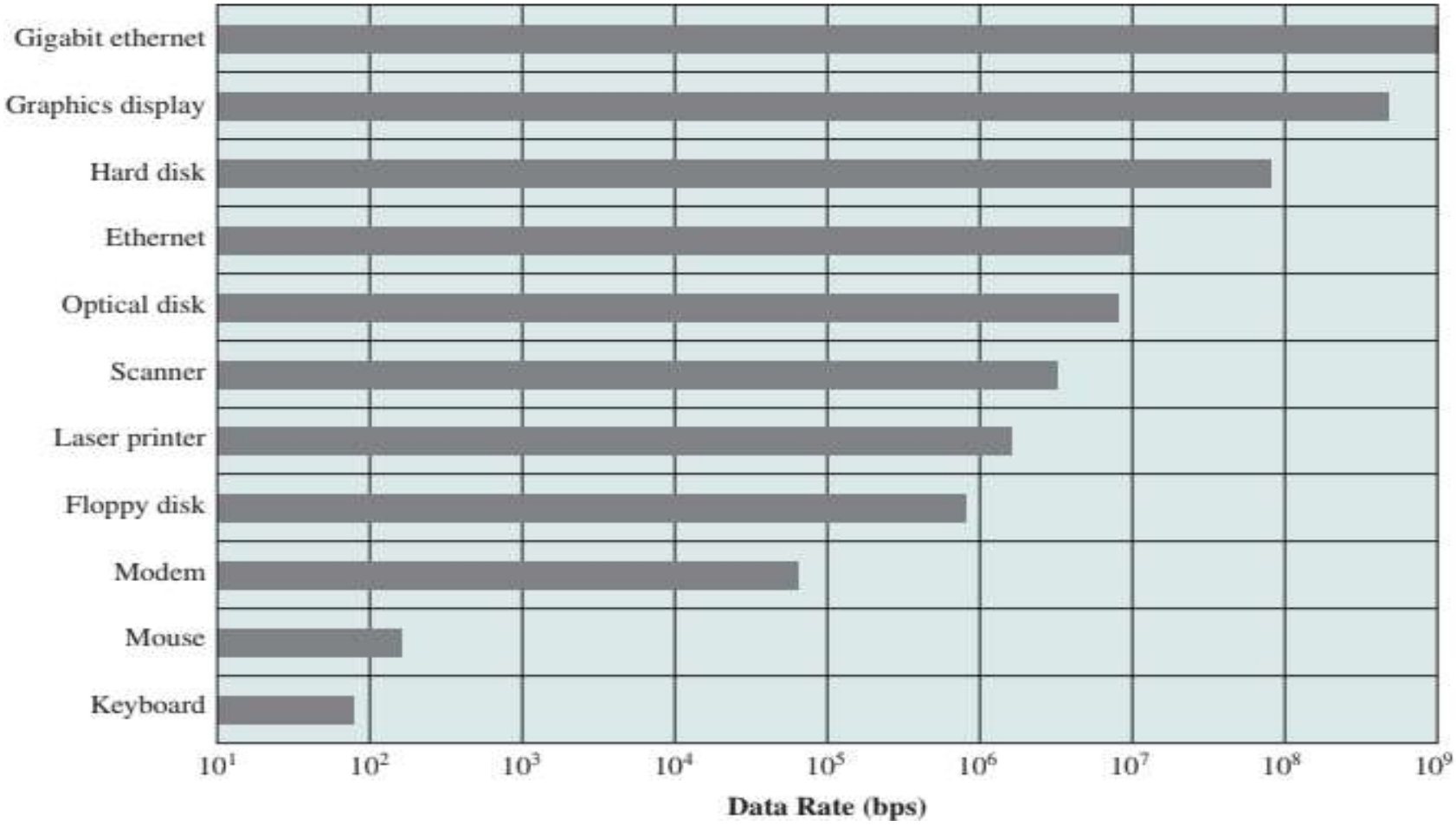
I/O DEVICES

Roughly grouped into three categories:

- **Human readable:** Suitable for communicating with the computer user. Examples include printers and terminals, the latter consisting of video display, keyboard, and perhaps other devices such as a mouse.
- **Machine readable:** Suitable for communicating with electronic equipment. Examples are disk drives, sensors, controllers, and actuators.
- **Communication:** Suitable for communicating with remote devices. Examples are digital line drivers and modems.

There are great differences across classes and even substantial differences within each class.
Among the key differences are the following:
• Data rate: Data rate is **how fast can a typical I/O device supply data to a computer?**

There may be differences of several orders of magnitude between the data transfer rates.Refer to the following diagram.


Gigabit ethernet
Graphics display
Hard disk
Ethernet
Optical disk
Scanner
Laser printer
Floppy disk
Modem
Mouse
Keyboard
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8
10^9
Data Rate (bps)

• **Application:** The use to which a device is put has an influence on the software and policies in the OS and supporting utilities. E.g

For example, a disk used for files requires the support of file management software. A disk used as a backing store for pages in a virtual memory scheme depends on the use of virtual memory hardware and software.

• **Complexity of control:** A printer requires a relatively simple control interface. A disk is much more complex. The effect of these differences on the OS is filtered to some extent 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 (e.g., terminal I/O) or in larger blocks (e.g., disk I/O).
- **Data representation:** Different data encoding schemes are used by different devices, including differences in character code and parity conventions.
- **Error conditions:** The nature of errors, the way in which they are reported, their consequences, and the available range of responses differ widely from one device to another.

ORGANIZATION OF THE I/O FUNCTION

Three techniques:

1. **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.

2. Interrupt-driven I/O:

The processor issues an I/O command on behalf of a process. There are then two possibilities.

If the I/O instruction from the process is nonblocking, then the processor continues to execute instructions from the process that issued the I/O command.

If the I/O instruction is blocking, then the next instruction that the processor executes is from the OS, which will put the current process in a blocked state and schedule another process.

3. 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.

The DMA unit is capable of mimicking the processor and, indeed, of taking over control of the system bus just like a processor. It needs to do this to transfer data to and from memory over the system bus. The DMA logic is shown in the diagram.

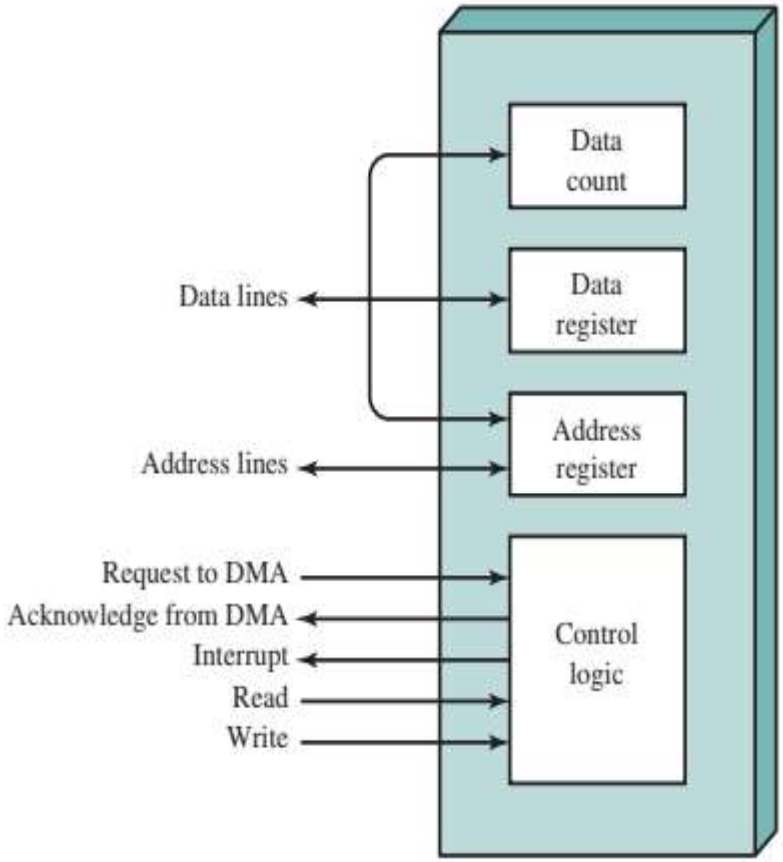


Figure 11.2 **Typical DMA Block Diagram**

The DMA technique works as follows. 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. Thus, the processor is involved only at the beginning and end of the transfer

OPERATING SYSTEM DESIGN ISSUES

Design Objectives

Two objectives are paramount in designing the I/O facility: **efficiency** and **generality**.

Efficiency is important because I/O operations often form a bottleneck in a computing system. Most I/O devices are extremely slow compared with main memory and the processor. One way to tackle this problem is multiprogramming, allows some processes to be waiting on I/O operations while another process is executing.

Thus, a major effort in I/O design has been schemes for improving the efficiency of the I/O. The area that has received the most attention, because of its importance, is disk I/O.

I/O BUFFERING : Need

Suppose that a user process wishes to read blocks of data from a disk one at a time, with each block having a length of 512 bytes. The data are to be **read** into a data area within the address space of the user process at virtual location 1000 to 1511.

The simplest way would be to execute an I/O command (something like Read_Block[1000, disk]) to the disk unit and then wait for the data to become available.

The waiting could either be busy waiting (continuously test the device status) or, more practically, process suspension on an interrupt.

There are two problems with this approach.

First, the program is hung up waiting for the relatively slow I/O to complete.

The second problem is that this approach to I/O interferes with swapping decisions by the OS.

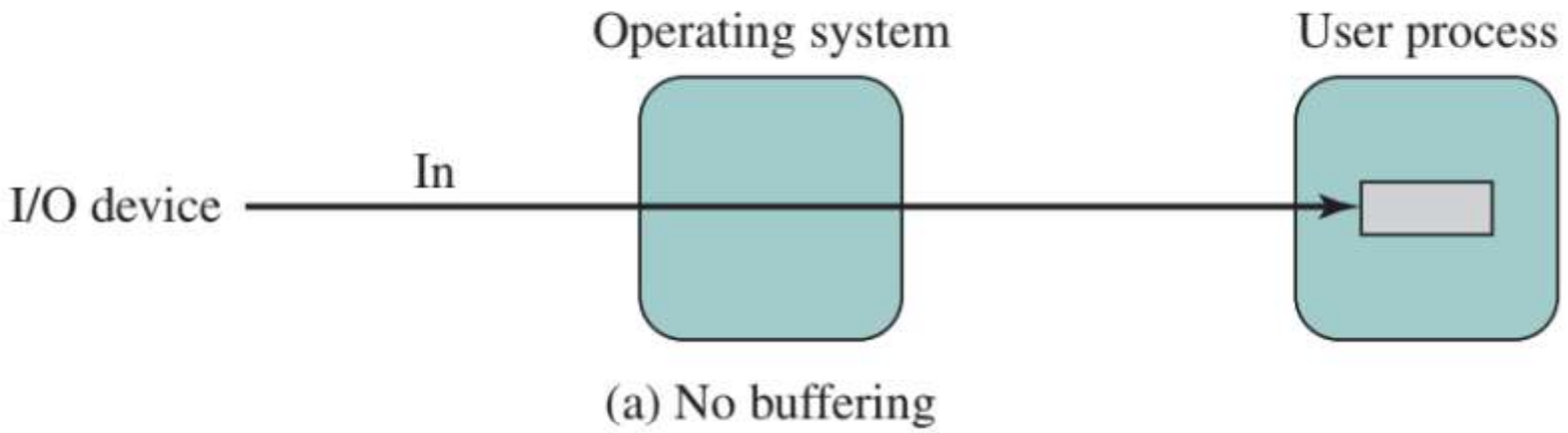
The same considerations apply to an **output** operation. If a block is being transferred from a user process area directly to an I/O module, then the process is blocked during the transfer and the process may not be swapped out.

To avoid these overheads and inefficiencies, it is sometimes convenient to perform input transfers in advance of requests being made and to perform output transfers some time after the request is made. This technique is known as buffering.

Single Buffer

The simplest type of support that the OS can provide is single buffering (As shown in the diagram)

When a user process issues an I/O request, the OS assigns a buffer in the system portion of main memory to the operation.



(a) No buffering

(b) Single buffering

This approach will generally provide a **speedup** compared to the lack of system buffering. The user process can be processing one block of data while the next block is being read in. The OS is able to swap the process out because the input operation is taking place in system memory rather than user process memory.

This technique does, **however, complicate the logic in the operating system. The OS must keep track of the assignment of system buffers to user processes. The swapping logic is also affected.**

Double Buffer

An improvement over single buffering can be had by assigning two system buffers to the operation (As shown in the next diagram).

A process now transfers data to (or from) one buffer while the operating system empties (or fills) the other. This technique is known as double buffering or buffer swapping .

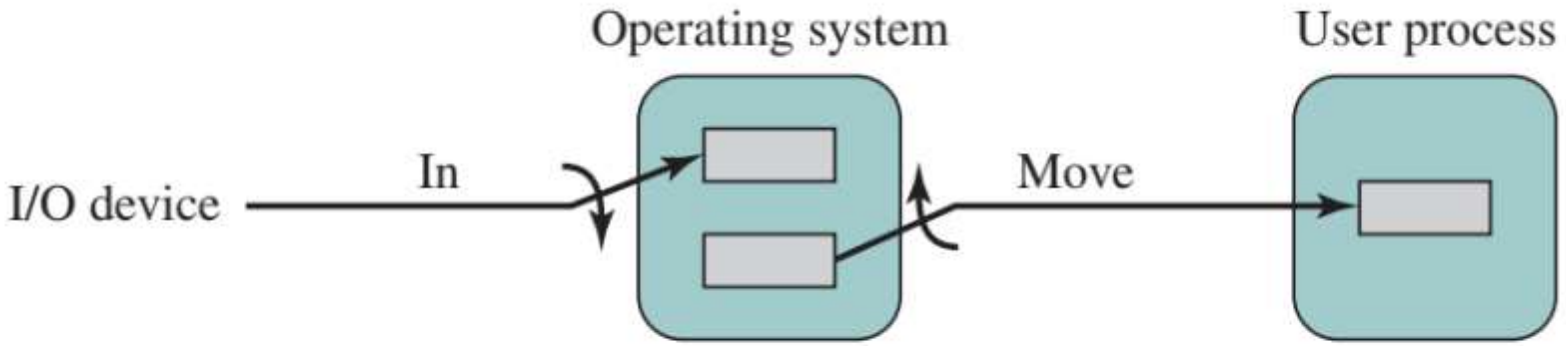
An improvement over single buffering is achieved. Again, this improvement comes at the cost of increased complexity.

Circular Buffer

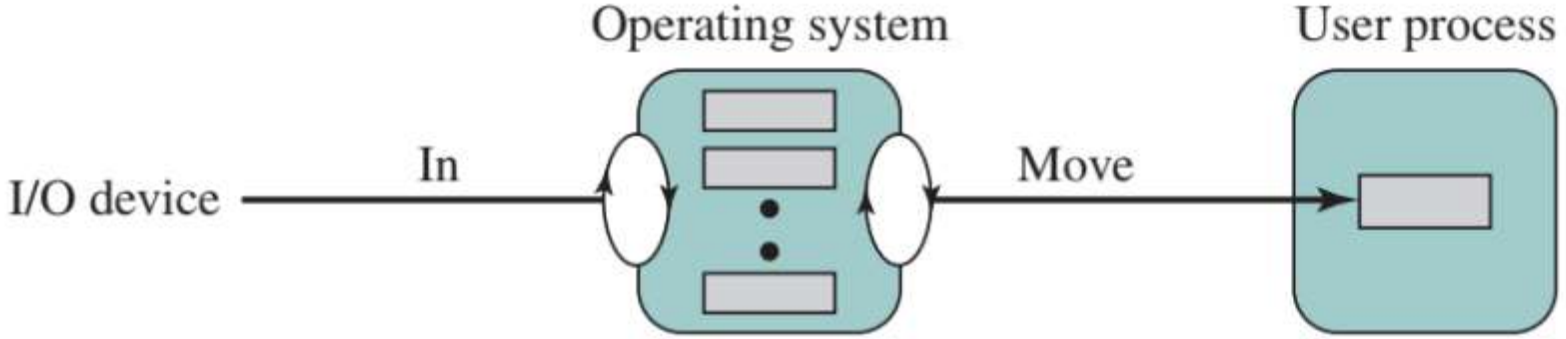
A double-buffer scheme should smooth out the flow of data between an I/O device and a process. If the performance of a particular process is the focus of our concern, then we would like for the I/O operation to be able to keep up with the process.

Double buffering may be inadequate if the process performs rapid bursts of I/O. In this case, the problem can often be alleviated by using more than two buffers.

When more than two buffers are used, the collection of buffers is itself referred to as a circular buffer , with each individual buffer being one unit in the circular buffer.



(c) Double buffering

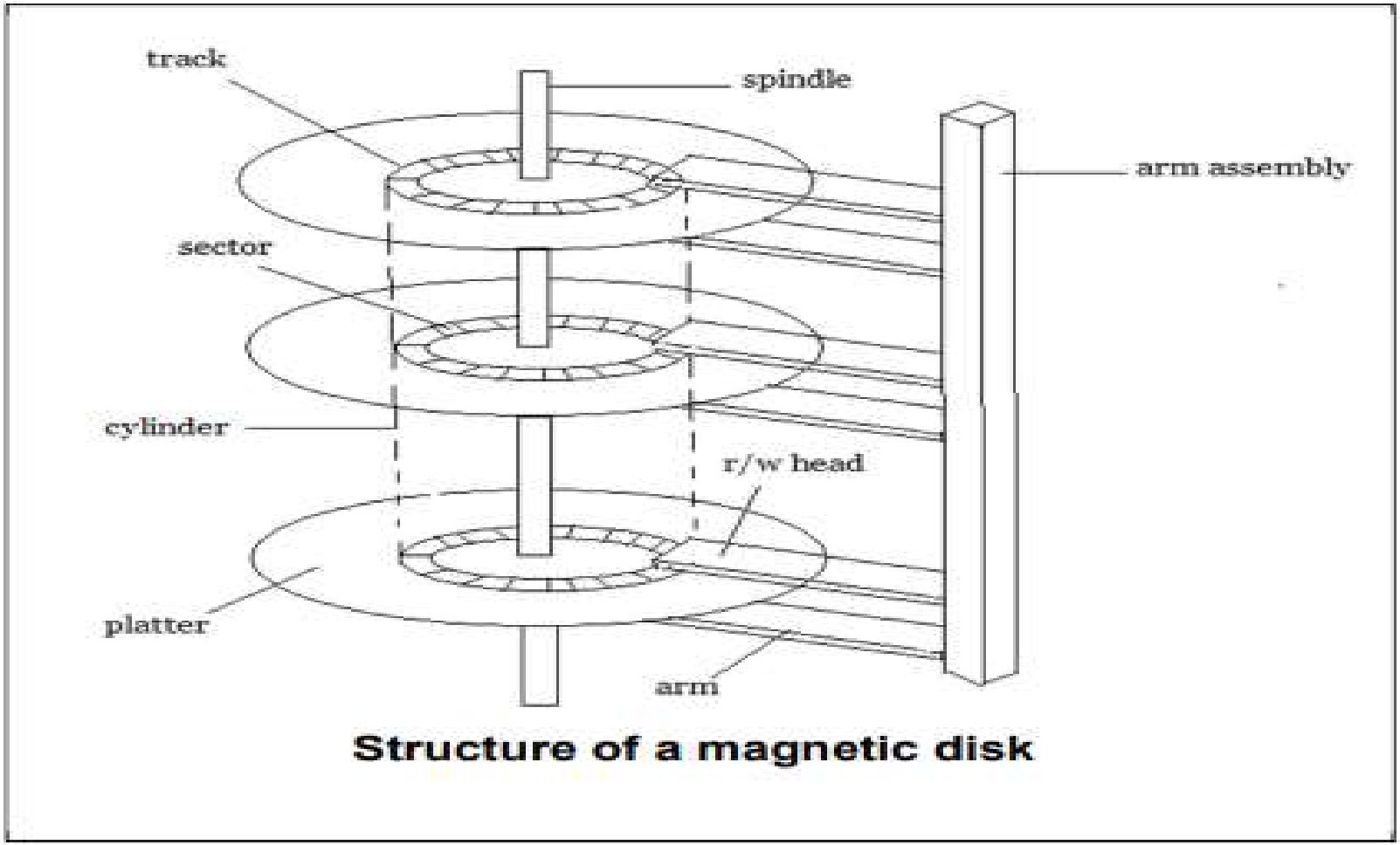


(d) Circular buffering

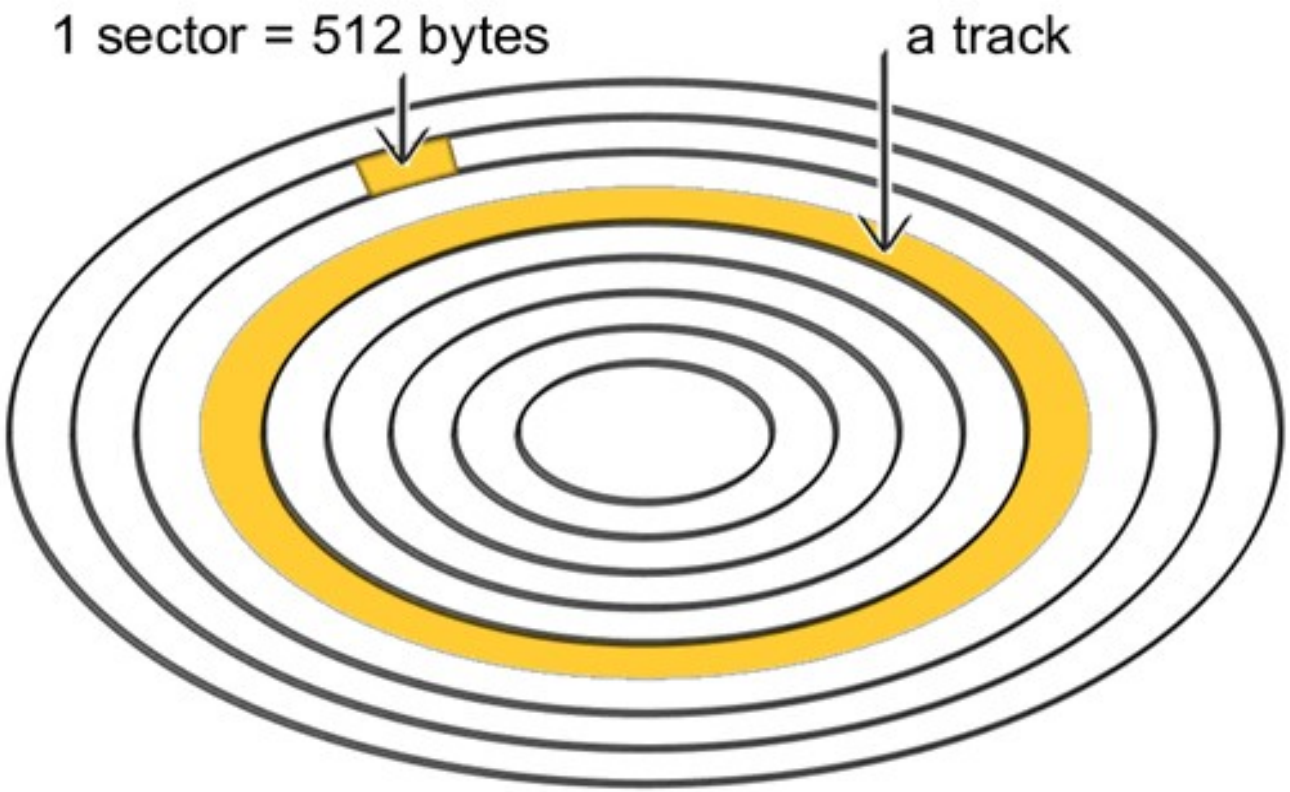
The Utility of Buffering

Buffering is a technique that smoothes out peaks in I/O demand. However, no amount of buffering will allow an I/O device to keep pace with a process indefinitely when the average demand of the process is greater than the I/O device can service. Even with multiple buffers, all of the buffers will eventually fill up and the process will have to wait after processing each chunk of data. However, **in a multiprogramming environment, when there is a variety of I/O activity and a variety of process activity to service, buffering is one tool that can increase the efficiency of the OS and the performance of individual processes.**

The read/write heads are the most expensive part of the hard disk. When the disk is switched off, the heads are parked in a special area, so that they will not be damaged during transport. In Fig. the cover has been removed from a hard disk, and you can see the uppermost arm with its read/write head.



Structure of a magnetic disk


1 sector = 512 bytes
a track

Tracks and sectors

Each hard disk plate is divided into tracks. Each track is subdivided into a number of sectors, the disk's smallest unit. A sector normally holds 512 bytes of data.

DISK SCHEDULING

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. **Disk scheduling is also known as I/O Scheduling.**

Disk Performance Parameters

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

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.**

Seek time

Seek time is the time required to move the disk arm to the required track.

That is, the time it takes to position the head at the track is known as **seek time** .

Rotational delay or Rotational latency

Once the track is selected, the disk controller waits until the appropriate sector rotates to line up with the head. The time it takes for the beginning of the sector to reach the head is known as rotational delay , or rotational latency.

Access time

The sum of the seek time, if any, and the rotational delay equals the **access time** , which is the time it takes to get into position to read or write.

Transfer time

Once the head is in position, the read or write operation is then performed as the sector moves under the head; this is the data transfer portion of the operation; the time required for the transfer is the transfer time.

Transfer time

The transfer time to or from the disk depends on the rotation speed of the disk in the following fashion:

$T = b / r * N$

where

T transfer time

b number of bytes to be transferred

N number of bytes on a track

r rotation speed, in revolutions per second

In addition to the access time and transfer time, there are several **queueing delays** normally associated with a disk I/O operation. When a process issues an I/O request, it must first wait in a queue for the device to be available. At that time, the device is assigned to the process. If the device shares a single I/O channel or a set of I/O channels with other disk drives, then there may be an additional wait for the channel to be available. At that point, the seek is performed to begin disk access.

Disk Scheduling Policies

Random scheduling.

Consider the typical situation in a multiprogramming environment, in which the OS maintains a queue of requests for each I/O device. So, for a single disk, there will be a number of I/O requests (reads and writes) from various processes in the queue. If we selected items from the queue in random order, then we can expect that the tracks to be visited will occur randomly, giving poor performance.

This random scheduling.

The performance of various scheduling algorithms for an example sequence of I/O requests. The vertical axis corresponds to the tracks on the disk. The horizontal axis corresponds to time or, equivalently, the number of tracks traversed.

We assume a disk with 200 tracks and that the disk request queue has random requests in it.

For each algorithm, we assume that the disk head is initially located at track 100.

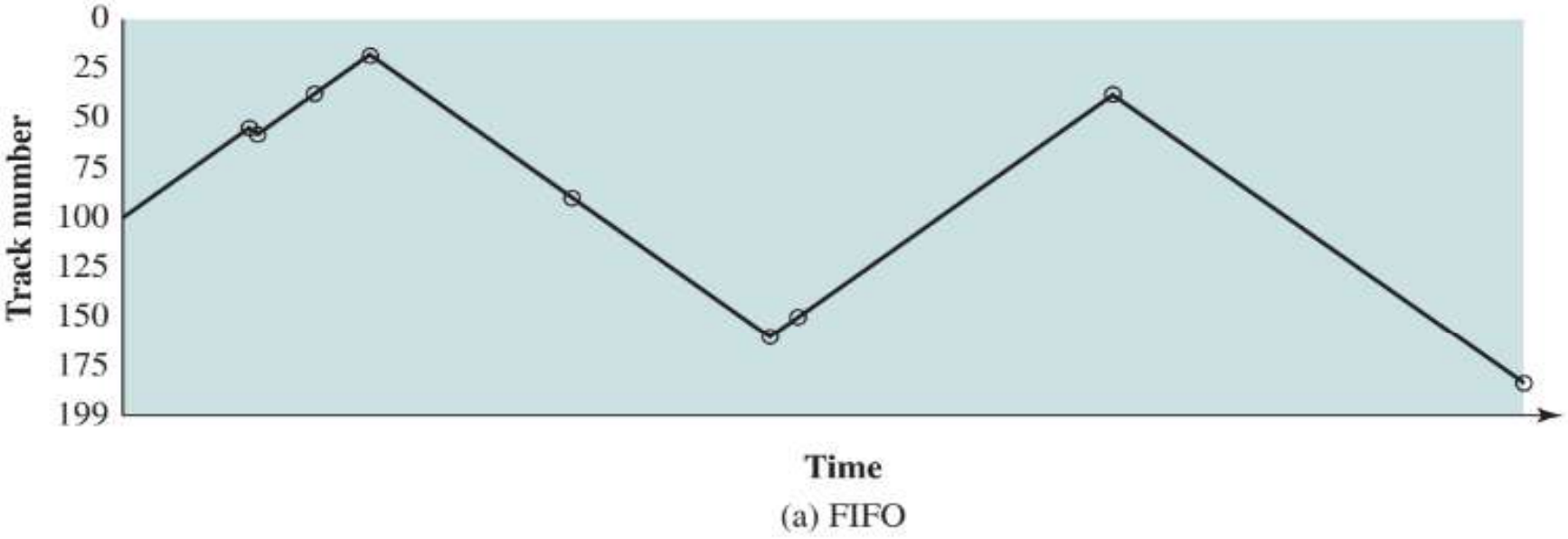
The requested tracks, in the order received by the disk scheduler, are 55, 58, 39, 18, 90, 160, 150, 38, 184.

FIRST IN FIRST OUT (FCFS)

The simplest form of scheduling is first-in-first-out (FIFO) scheduling, which processes items from the queue in sequential order.

(a) FIFO (starting at track 100)

Next track accessed	Number of tracks traversed
55	45
58	3
39	19
18	21
90	72
160	70
150	10
38	112
184	146
Average seek length	55.3



(a) FIFO

FIRST IN FIRST OUT (FIFO)

Advantage:

This strategy has the advantage of being fair, because every request is honored and the requests are honored in the order received.

Disadvantage:

With FIFO, if there are only a few processes that require access and if many of the requests are to clustered file sectors, then we can hope for good performance.

However, this technique will often approximate random scheduling in performance, if there are many processes competing for the disk.

Performance gets affected by number of processes.

PRIORITY

With a system based on priority (PRI), the control of the scheduling is outside the control of disk management software. Such **an approach is not intended to optimize disk utilization but to meet other objectives within the OS**. Often short batch jobs and interactive jobs are given higher priority than longer jobs that require longer computation. This allows a lot of short jobs to be flushed through the system quickly and may provide good interactive response time. However, longer jobs may have to wait excessively long times.

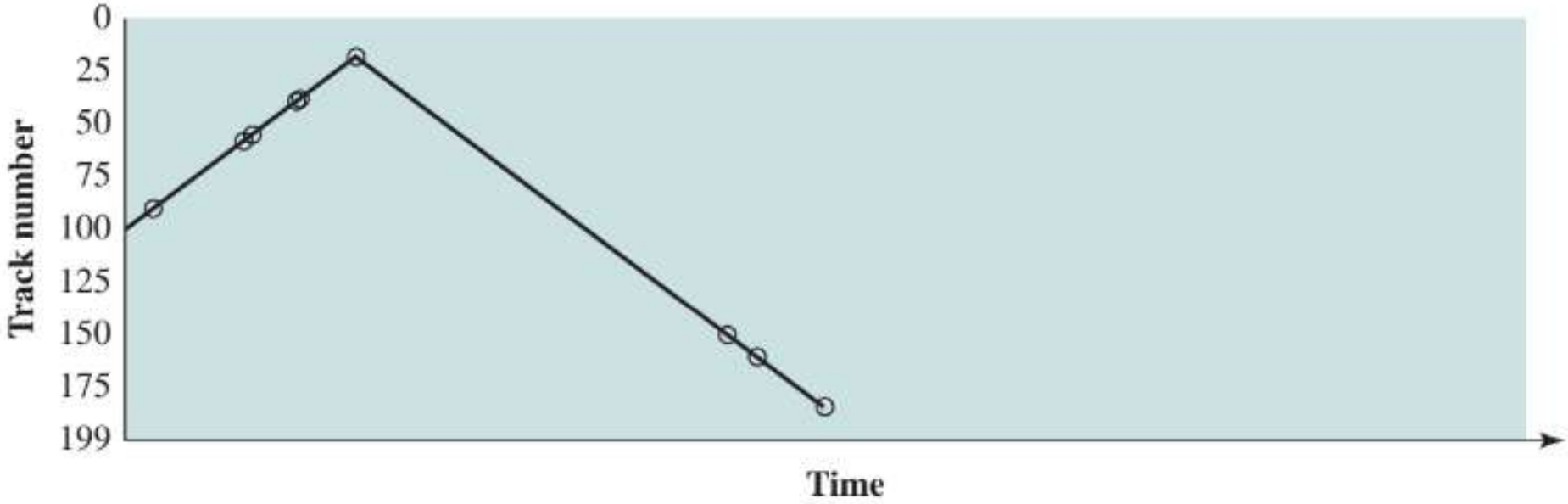
Furthermore, such a policy could lead to countermeasures on the part of users, who split their jobs into smaller pieces to beat the system. **This type of policy tends to be poor for database systems.**

SHORTEST SERVICE TIME FIRST

The shortest-service-time-first (SSTF) policy **is to select the disk I/O request that requires the least movement of the disk arm from its current position.** Thus, we always choose to incur the minimum seek time. Of course, always choosing the minimum seek time does not guarantee that the average seek time over a number of arm movements will be minimum. However, this should provide better performance than FIFO. Because the arm can move in two directions, a random tie-breaking algorithm may be used to resolve cases of equal distances.

(b) SSTF (starting at track 100)

Next track accessed	Number of tracks traversed
90	10
58	32
55	3
39	16
38	1
18	20
150	132
160	10
184	24
Average seek length	27.5



(b) SSTF

Above diagram and table show the performance of SSTF on the same example as was used for FIFO. The first track accessed is 90, because this is the closest requested track to the starting position. The next track accessed is 58 because this is the closest of the remaining requested tracks to the current position of 90. Subsequent tracks are selected accordingly.

SCAN With the exception of FIFO, all of the policies described so far can leave some request unfulfilled until the entire queue is emptied. That is, there may always be new requests arriving that will be chosen before an existing request.

A simple alternative that prevents this sort of starvation is the **SCAN algorithm, also known as the elevator algorithm because it operates much the way an elevator does.**

The diagram and the table illustrate the SCAN policy. Assuming that the initial direction is of increasing track number, then the first track selected is 150, since this is the closest track to the starting track of 100 in the increasing direction.

SCAN With the exception of FIFO, all of the policies described so far can leave some request unfulfilled until the entire queue is emptied. That is, there may always be new requests arriving that will be chosen before an existing request.

A simple alternative that prevents this sort of starvation is the **SCAN algorithm, also known as the elevator algorithm because it operates much the way an elevator does.**

The diagram and the table illustrate the SCAN policy. Assuming that the initial direction is of increasing track number, then the first track selected is 150, since this is the closest track to the starting track of 100 in the increasing direction.

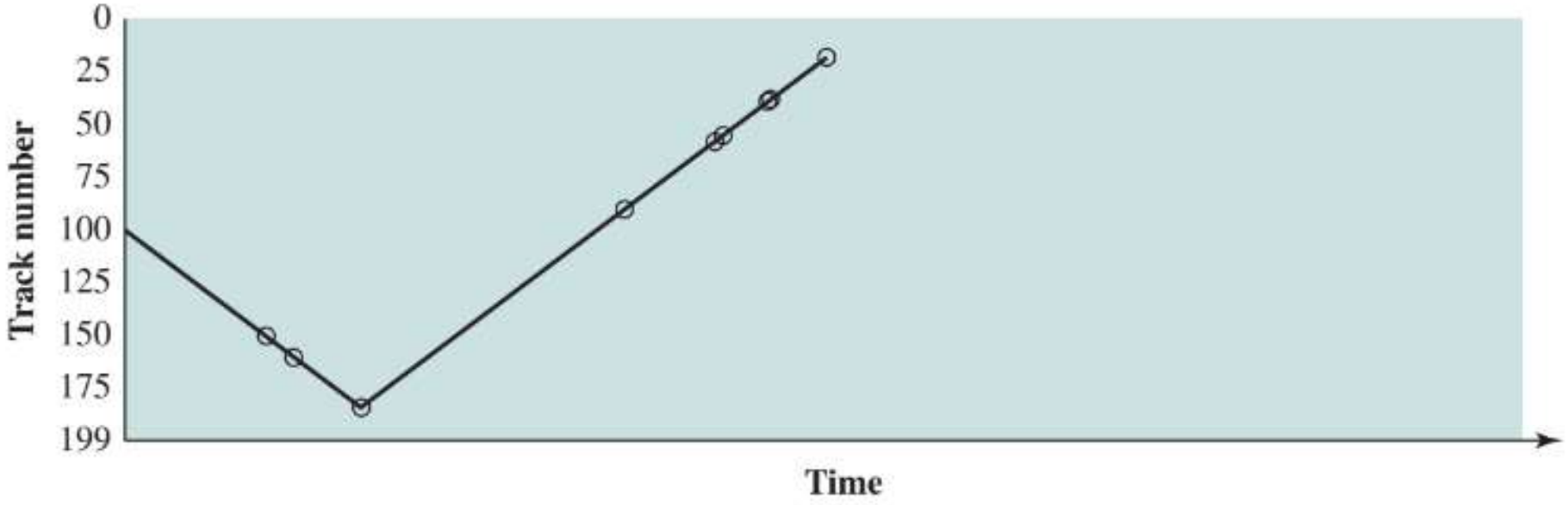
SCAN With the exception of FIFO, all of the policies described so far can leave some request unfulfilled until the entire queue is emptied. That is, there may always be new requests arriving that will be chosen before an existing request.

A simple alternative that prevents this sort of starvation is the **SCAN algorithm, also known as the elevator algorithm because it operates much the way an elevator does.**

The diagram and the table illustrate the SCAN policy. Assuming that the initial direction is of increasing track number, then the first track selected is 150, since this is the closest track to the starting track of 100 in the increasing direction.

(c) SCAN (starting at track 100, in the direction of increasing track number)

Next track accessed	Number of tracks traversed
150	50
160	10
184	24
90	94
58	32
55	3
39	16
38	1
18	20
Average seek length	27.8



(c) SCAN

Look

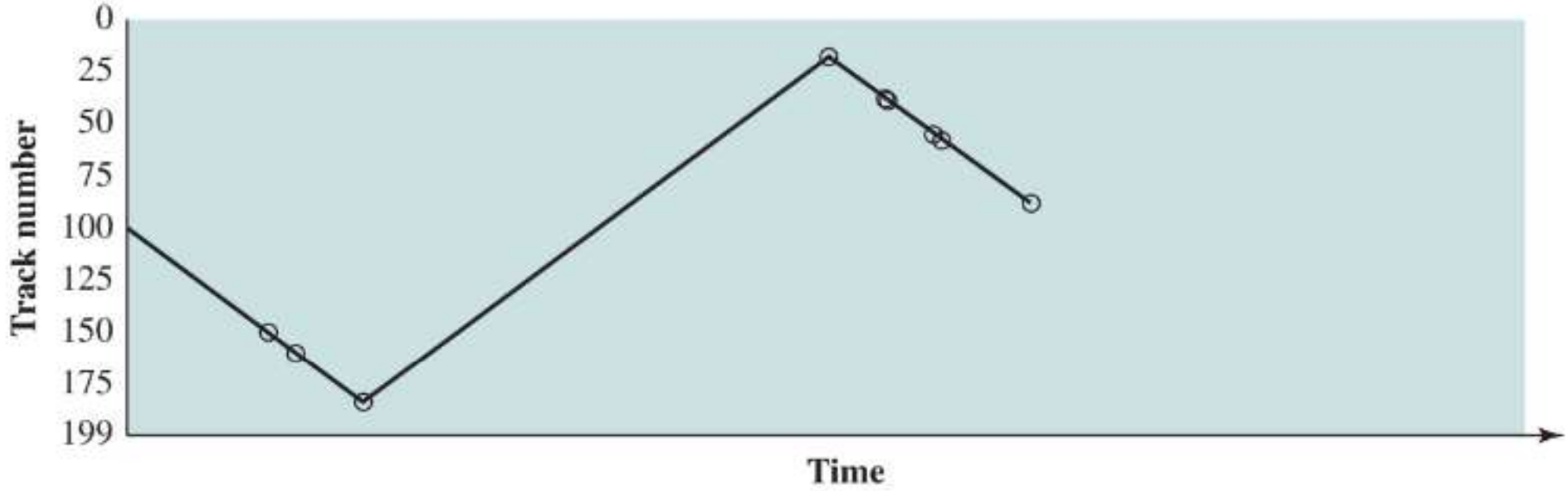
With SCAN, the arm is required to move in one direction only, satisfying all outstanding requests en route, until it reaches the last track in that direction or until there are no more requests in that direction. **This refinement is referred to as the LOOK policy.** The service direction is then reversed and the scan proceeds in the opposite direction, again picking up all requests in order.

C-SCAN The C-SCAN (circular SCAN) policy restricts scanning to one direction only. Thus, 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. **This reduces the maximum delay experienced by new requests.**

The diagram and the table illustrate C-SCAN behavior. In this case the first three requested tracks encountered are 150, 160, and 184. Then the scan begins starting at the lowest track number, and the next requested track encountered is 18.

(d) C-SCAN (starting at track 100, in the direction of increasing track number)

Next track accessed	Number of tracks traversed
150	50
160	10
184	24
18	166
38	20
39	1
55	16
58	3
90	32
Average seek length	35.8



(d) C-SCAN

File Management

In most applications, the file is the central element. With the exception of real-time applications and some other specialized applications, the input to the application is by means of a file; and in virtually all applications, output is saved in a file for long term storage and for later access by the user and by other programs.
Files have a life outside of any individual application that uses them for input and/or output. Users wish to be able to access files, save them, and maintain the integrity of their contents.

File Management

To aid in these objectives, virtually all operating systems provide file management systems. Typically, a file management system consists of system utility programs that run as privileged applications. However, at the very least, a file management system needs special services from the operating system; at the most, **the entire file management system is considered part of the operating system.**

Files and File Systems

From the user's point of view, one of the most important parts of an operating system is the file system. **The file system provides the resource abstractions typically associated with secondary storage. The file system permits users to create data collections, called files.**

File Structure

Four terms are in common use when discussing files:

- **Field**
- **Record**
- **File**
- **Database**

A **field** is the basic element of data. An individual field contains a single value, such as an employee's last name, a date, or the value of a sensor reading. It is characterized by its length and data type (e.g., ASCII string, decimal).

A **record** is a collection of related fields that can be treated as a unit by some application program. For example, an employee record would contain such fields as name, social security number, job classification, date of hire, and so on.

A **file** is a collection of similar records. The file is treated as a single entity by users and applications and may be referenced by name. Files have file names and may be created and deleted. Access control restrictions usually apply at the file level. That is, in a shared system, users and programs are granted or denied access to entire files.

A **database** is a collection of related data. A database may contain all of the information related to an organization or project, such as a business or a scientific study. The database itself consists of one or more types of files.
Usually, there is a separate database management system that is independent of the operating system, although that system may make use of some file management programs.

File Management Systems

A file management system is that set of system software that provides services to users and applications in the use of files. Typically, the only way that a user or application may access files is through the file management system.

For an interactive, general-purpose system, the following constitute a minimal set of requirements:

1. Each user should be able to create, delete, read, write, and modify files.
2. Each user may have controlled access to other users' files.
3. Each user may control what types of accesses are allowed to the user's files.
4. Each user should be able to restructure the user's files in a form appropriate to the problem.
5. Each user should be able to move data between files.
6. Each user should be able to back up and recover the user's files in case of damage.
7. Each user should be able to access his or her files by name rather than by numeric identifier.

FILE SYSTEM ARCHITECTURE

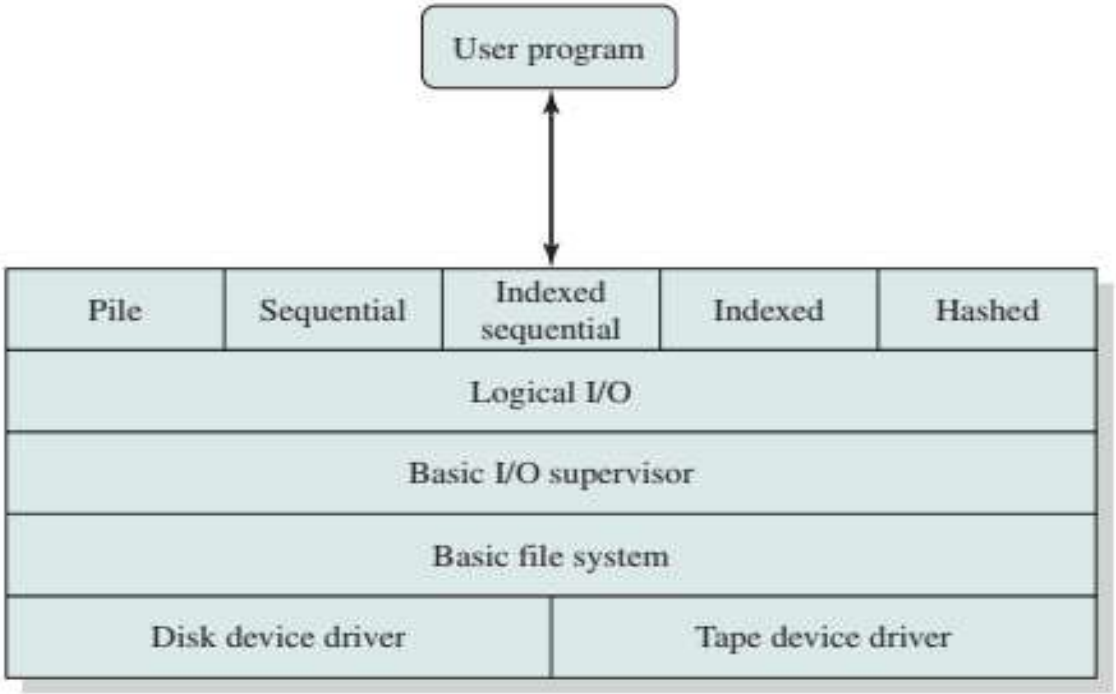


Figure 12.1 File System Software Architecture

FILE SYSTEM ARCHITECTURE

At the lowest level, **device drivers** communicate directly with peripheral devices or their controllers or channels.

The next level is referred to as the **basic file system , or the physical I/O level**. This is the primary interface with the environment outside of the computer system.

The **basic I/O supervisor** is responsible for all file I/O initiation and termination.

Logical I/O enables users and applications to access records.

The level of the **file system** closest to the user is often termed the access method. It provides a standard interface between applications and the file systems and devices that hold the data.

FILE ORGANIZATION AND ACCESS

The term **file organization to refer to the logical structuring of the records as determined by the way in which they are accessed.** The physical organization of the file on secondary storage depends on the blocking strategy and the file allocation strategy.

The five organizations are as follows:

- The pile
- The sequential file
- The indexed sequential file
- The indexed file
- The direct, or hashed, file

The Pile

The least-complicated form of file organization may be termed the pile . **Data are collected in the order in which they arrive.** Each record consists of one burst of data. The purpose of the pile is simply to accumulate the mass of data and save it. Records may have different fields, or similar fields in different orders.

The Sequential File

The most common form of file structure is the sequential file. In this type of file, a **fixed format** is used for records. All records are of the same length, consisting of the same number of fixed-length fields in a particular order. Because the length and position of each field are known, only the values of fields need to be stored; the field name and length for each field are attributes of the file structure.

For interactive applications that involve queries and/or updates of individual records, the sequential file provides poor performance.

The Indexed Sequential File

A popular approach to overcoming the disadvantages of the sequential file is the indexed sequential file. **The indexed sequential file maintains the key characteristic of the sequential file: Records are organized in sequence based on a key field.**

Two features were added:

1. an index to the file to support random access,
2. an overflow file **-** The overflow file is similar to the log file

The Indexed File

The indexed sequential file retains one limitation of the sequential file: Effective processing is limited to that which is based on a single field of the file.

In some applications, the flexibility of efficiently searching by various attributes is desirable. To achieve this flexibility, a structure is needed that employs multiple indexes, one for each type of field that may be the subject of a search. In the general indexed file, the concept of sequentiality and a single key are abandoned. Records are accessed only through their indexes.

The Direct or Hashed File

The direct, or hashed, file exploits the capability found on disks to access directly any block of a known address.

FILE SYSTEM SECURITY

Following successful log-on, the user has been granted access to one or a set of hosts and applications. This is generally not sufficient for a system that includes sensitive data in its database. Through the user–access control procedure, a user can be identified to the system. Associated with each user, there can be a profile that specifies permissible operations and file accesses. The operating system can then enforce rules based on the user profile.

Operating System Security

What is Operating System Security?

- The process of ensuring OS **availability**, **confidentiality**, **integrity** is known as operating system security.
- OS security refers to the processes or measures taken to protect the operating system from dangers, including viruses, worms, malware, and remote hacker intrusions.
- Operating system security comprises all preventive-control procedures that protect any system assets that could be stolen, modified, or deleted if OS security is breached.

Security refers to providing safety for computer system resources like software, CPU, memory, disks, etc. It can protect against all **threats**, including **viruses** and **unauthorized access**. It can be enforced by assuring the operating system's **integrity**, **confidentiality**, and **availability**. If an illegal user runs a computer application, the computer or data stored may be seriously damaged.

Confidentiality is a set of rules that **limits access to information**,
Integrity is the **assurance that the information is trustworthy and accurate**.
Availability is a **guarantee of reliable access to the information by authorized people**.

System security may be threatened through two violations, and these are as follows:

1.Threat: A program that has the potential to harm the system seriously.

2.Attack: A breach of security that allows unauthorized access to a resource.

There are **two types of security breaches** that can harm the system: **malicious and accidental.** Malicious threats are a type of destructive computer code or web script that is designed to cause system vulnerabilities that lead to back doors and security breaches. On the other hand, Accidental Threats are comparatively easier to protect against.

Types of Threats

There are mainly two types of threats that occur. These are as follows:

Program threats

The operating system's processes and kernel carry out the specified task as directed. Program Threats occur when a user program causes these processes to do malicious operations.

1.Virus

A virus may replicate itself on the system. Viruses are extremely dangerous and can modify/delete user files as well as crash computers.

2. Trojan Horse

This type of application captures user login credentials. It stores them to transfer them to a malicious user who can then log in to the computer and access system resources.

3. Logic Bomb

A logic bomb is a situation in which software only misbehaves when particular criteria are met; otherwise, it functions normally.

System Threats

System threats are described as the misuse of system services and network connections to cause user problems.

1. **Port Scanning:** It is a fully automated process that includes connecting to a specific port via TCP/IP.
2. **Worm:** The worm is a process that can choke a system's performance by exhausting all system resources.
3. **Denial of service:** Attacks usually prevents users from legitimately using the system

How to ensure Operating System Security?

There are various ways to ensure operating system security. These are as follows:

1. **Authentication and One Time passwords**
2. **Firewalls: Firewalls are essential for monitoring all incoming and outgoing traffic.**
3. **Physical Security**

Operating System Security Policies and Procedures

OS security policies and procedures are developed and implemented to ensure that you must first determine which assets, systems, hardware, and date are the most vital to your organization. Once that is completed, a policy can be developed to secure and safeguard them properly.

There are various techniques. Some of them are as follows:

1. Installing and updating anti-virus software
2. Ensure the systems are patched or updated regularly
3. Implementing user management policies to protect user accounts and privileges.
4. Installing a firewall and ensuring that it is properly set to monitor all incoming and outgoing traffic.