Unit VI
IO Management and Disk Scheduling

And

File Management

Contents

- File organization
- File directories
- Record blocking
- Secondary storage management
- I/O devices
- i/O buffering
- Disk Scheduling

File Management

- File management system consists of system utility programs that run as privileged applications.
- Input to applications is by means of a file.
- Output is saved in a file for long-term storage.
- At the very least, a file management system needs special services from OS, at the most the FMS is considered part of the OS

File System Properties

- Long-term existence: on secondary storage device
- Sharable between processes: by names n associated access permissions
- Structure: have internal structure, can be organized into hierarchical or more complex structure

File Operations

- Acts as a means to store data n have collection of functions that can be performed on them.
- Create
- Delete
- Open
- Close
- Read
- Write

Terms Used with Files

- Field
  - Basic element of data
  - Contains a single value
  - Characterized by its length and data type
  - Fixed/variable length
- Record
  - Collection of related fields treated as a unit
  - Fixed/variable length
    - Example: employee record

Terms Used with Files

- File
  - Collection of similar records
  - Treated as a single entity by user/application
  - Have file names
  - Access control restriction applied at file level
  - In some system , Access control restriction is applied at record/filed level also.
- Database
  - Collection of related data
  - Relationships exist among elements
  - Ex: Student database

Typical Operations

- Retrieve_All : displaying summery; similar to sequential processing
- Retrieve_One:Intereactive,transaction oriented appl.
- Retrieve_Next: Intereactive appl.for searching/filling any field
- Retrieve_Previous:Similar to Retrieve_Next:
- Insert_One:
- Delete_One :
- Update_One :
- Retrieve_Few :

File Management Systems

- Set of system softwares that provides services to users and applications in the use of files.
- The way a user of application may access files
- Programmer does not need to develop file management software

Objectives of File Management System

- Meet the data management needs and requirements of the user
- Guarantee that the data in the file are valid
- Optimize performance (System-Throughput, User-response time)
- Provide I/O support for a variety of storage device types
- Minimize or eliminate the potential for lost or destroyed data
- Provide a standardized set of I/O interface routines to use processes
- Provide I/O support for multiple users

Minimal Set of Requirements for interactive & general purpose system

- Each user should be able to create, delete, read, write and modify files
- Each user may have controlled access to other users' files
- Each user may control what type of accesses are allowed to the users' files
- Each user should be able to restructure the user's files in a form appropriate to the problem
- Each user should be able to move data between files
- Each user should be able to back up and recover the user's files in case of damage
- Each user should be able to access the user's files by using symbolic names

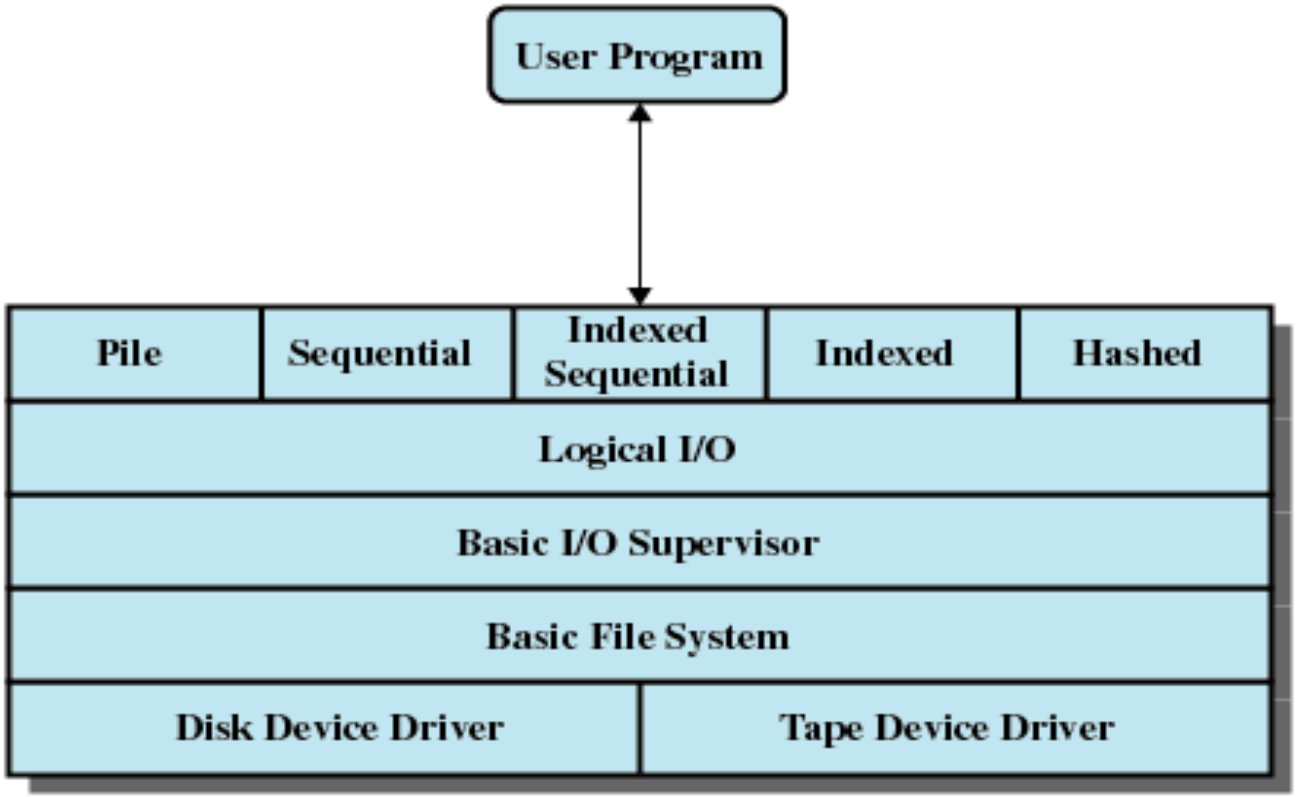


Figure 12.1 File System Software Architecture

Device Drivers

- Lowest level
- Communicates directly with peripheral devices/channels
- Responsible for starting I/O operations on a device &Processing the completion of an I/O request
- Typical devices controlled are disk & tape drives.
- Usually considered as part of OS

Basic File System

- Basic file system/Physical I/O
- Primary interface with the environment outside of computer
- Deals with exchanging blocks of data with tape/disk system
- Concerned with the placement of blocks on secondary storage
- Concerned with buffering blocks in main memory
- Considered as part of OS

Basic I/O Supervisor

- Responsible for file I/O initiation and termination
- Control structures are maintained that deal with
  - Device I/O,
  - Scheduling,
  - File status
- Concerned with selection of the device on which file I/O is to be performed
- Concerned with scheduling access to optimize performance
- Part of the operating system

Logical I/O

- Enables users and applications to access records
- Basic file system was dealing with block n it deals with records
- Provides general-purpose record I/O capability
- Maintains basic data about file

Access Method

- Closest to the user
- Provides a standard interface between applications and the file systems and devices that hold the data
- Reflect different file structures
- Access method varies depending on the ways to access and process data for the device.

File Management Functions

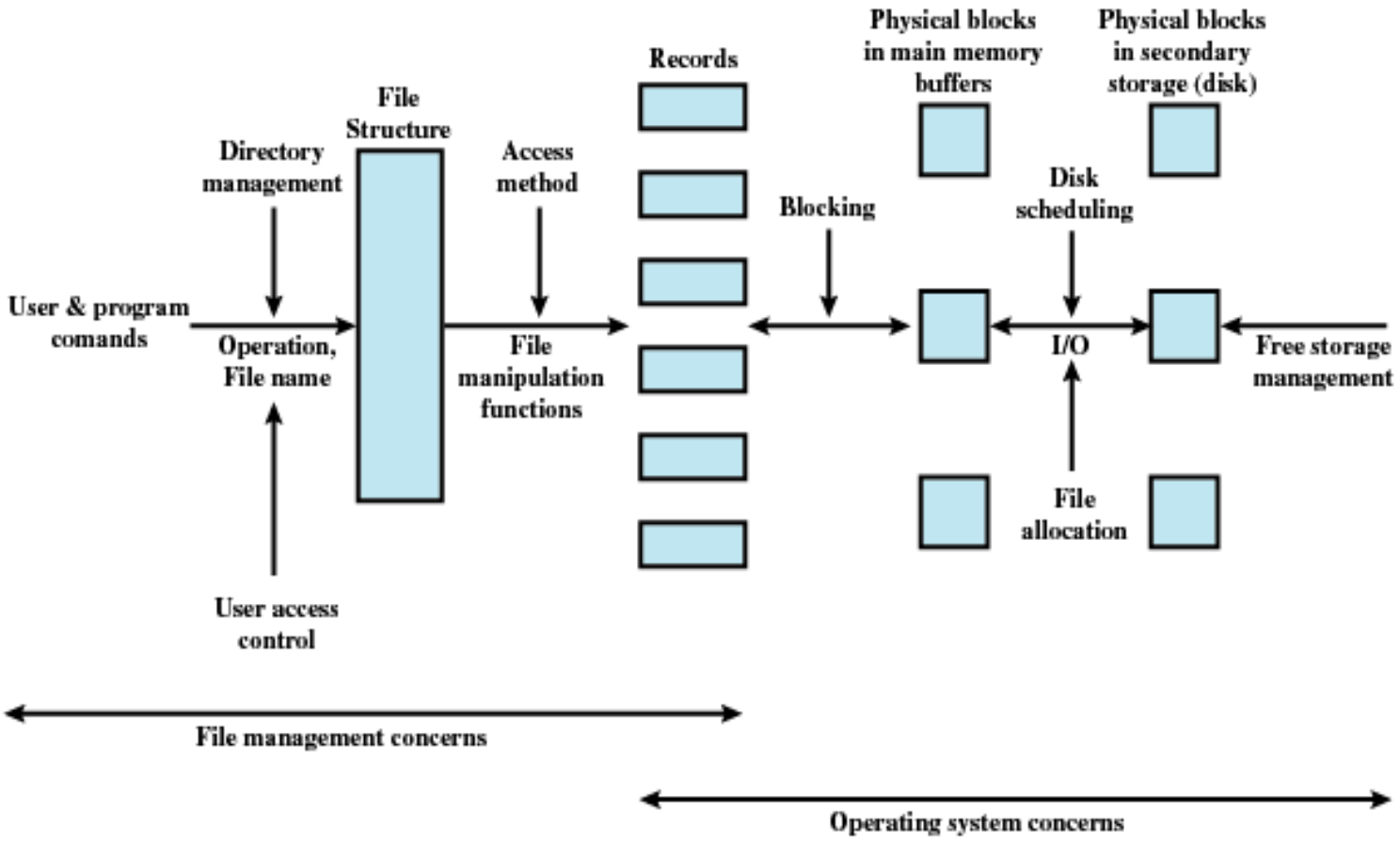


Figure 12.2 Elements of File Management

File Management Functions

- Identify and locate a selected file
- Use a directory to describe the location of all files plus their attributes
- On a shared system describe user access control
- Blocking for access to files
- Allocate files to free blocks
- Manage free storage for available blocks

Contents

- File Management Concept
- File sharing
- File organization
- **File directories**
- Record blocking
- i/O buffering
- Disk Scheduling

File Directories

- Contains information about files
  - Attributes
  - Location
  - Ownership
- Directory itself is a file owned by the operating system
- Provides mapping between file names and the files themselves

Directory Elements: Basic Information

- File Name
  - Name as chosen by creator (user or program).
  - Must be unique within a specific directory.
- File type:text,binary
- File Organisation
  - For systems that support different organizations

Directory Elements: Address Information

- Volume
  - Indicates device on which file is stored
- Starting Address starting physical address on secondary storage
- Size Used
  - Current size of the file in bytes, words, or blocks
- Size Allocated
  - The maximum size of the file

Directory Elements: Access Control Information

- Owner
  - The owner may be able to grant/deny access to other users and to change these privileges.
- Access Information
  - May include the user's name and password for each authorized user.
- Permitted Actions
  - Controls reading, writing, executing, transmitting over a network

Directory Elements: Usage Information

- Date Created
- Identity of Creator
- Date Last Read Access
- Identity of Last Reader
- Date Last Modified
- Identity of Last Modifier
- Date of Last Backup
- Current Usage
  - Current activity, locks, etc

Basic Information	
File Name	Name as chosen by creator (user or program). Must be unique within a specific directory
File Type	For example: text, binary, load module, etc.
File Organization	For systems that support different organizations
Address Information	
Volume	Indicates device on which file is stored
Starting Address	Starting physical address on secondary storage (e.g., cylinder, track, and block number on disk)
Size Used	Current size of the file in bytes, words, or blocks
Size Allocated	The maximum size of the file

Access Control Information	
Owner	User who is assigned control of this file. The owner may be able to grant/deny access to other users and to change these privileges.
Access Information	A simple version of this element would include the user's name and password for each authorized user.
Permitted Actions	Controls reading, writing, executing, and transmitting over a network
Usage Information	
Date Created	When file was first placed in directory
Identity of Creator	Usually but not necessarily the current owner
Date Last Read Access	Date of the last time a record was read
Identity of Last Reader	User who did the reading
Date Last Modified	Date of the last update, insertion, or deletion
Identity of Last Modifier	User who did the modifying
Date of Last Backup	Date of the last time the file was backed up on another storage medium
Current Usage	Information about current activity on the file, such as process or processes that have the file open, whether it is locked by a process, and whether the file has been updated in main memory but not yet on disk

Operations Performed on a Directory

- A directory system should support a number of operations including:
  - Search
  - Create files
  - Deleting files
  - Listing directory
  - Updating directory

Directory Organizations

- Simple
- Two level scheme
- Hierarchical or tree structured

Simple Structure for a Directory

- List of entries, one for each file
- Sequential file with the name of the file serving as the key
- Provides no help in organizing the files
- Forces user to be careful not to use the same name for two different files
- Problems

  Flat organization

  No logical grouping, no access control

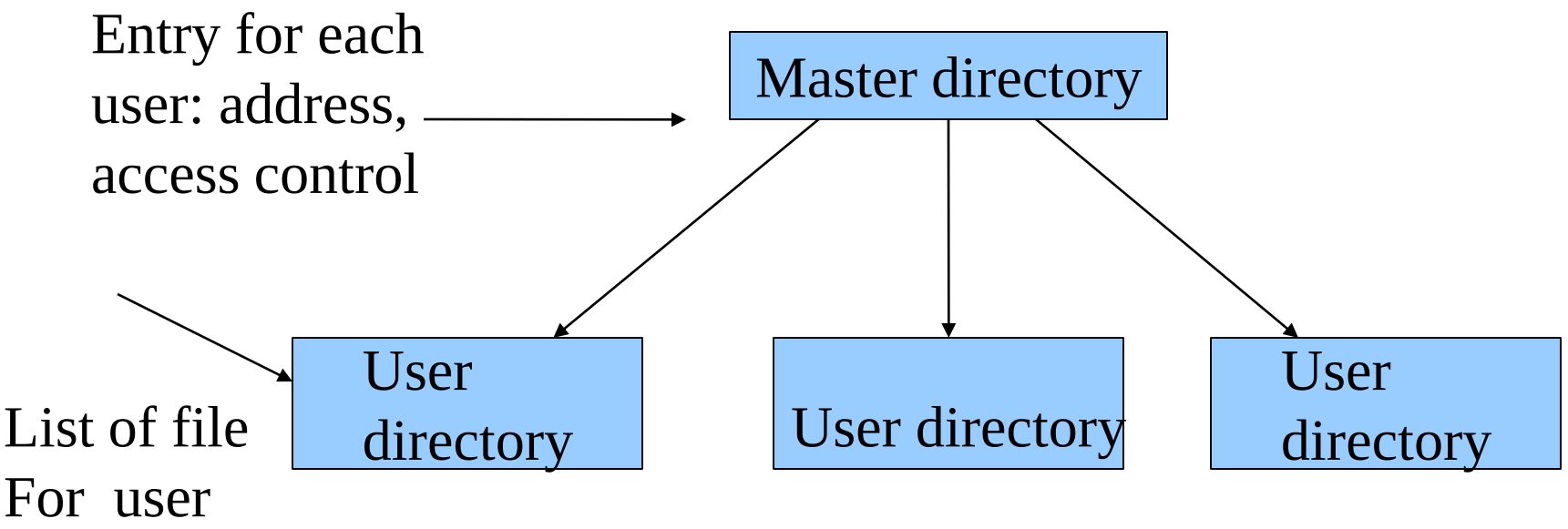
  Two files cannot have the same name

  Serious concern on shared user systems

Two-level Scheme for a Directory

- One directory for each user and a master directory
- Master directory contains entry for each user
  - Provides address and access control information
- Each user directory is a simple list of files for that user
- File name need to be unique only within a user directory
- System can easily enforce access restriction on directories
- Still provides no help in structuring collections of files

Two-level Scheme for a Directory



Hierarchical, or Tree-Structured Directory

- Master directory with user directories underneath it
- Each user directory may have subdirectories and files as entries
- Files can be located by following a path from the root, or master, directory down various branches
  - This is the pathname for the file
- Can have several files with the same file name as long as they have unique path names
- Current directory is the working directory
- Files are referenced relative to the working directory

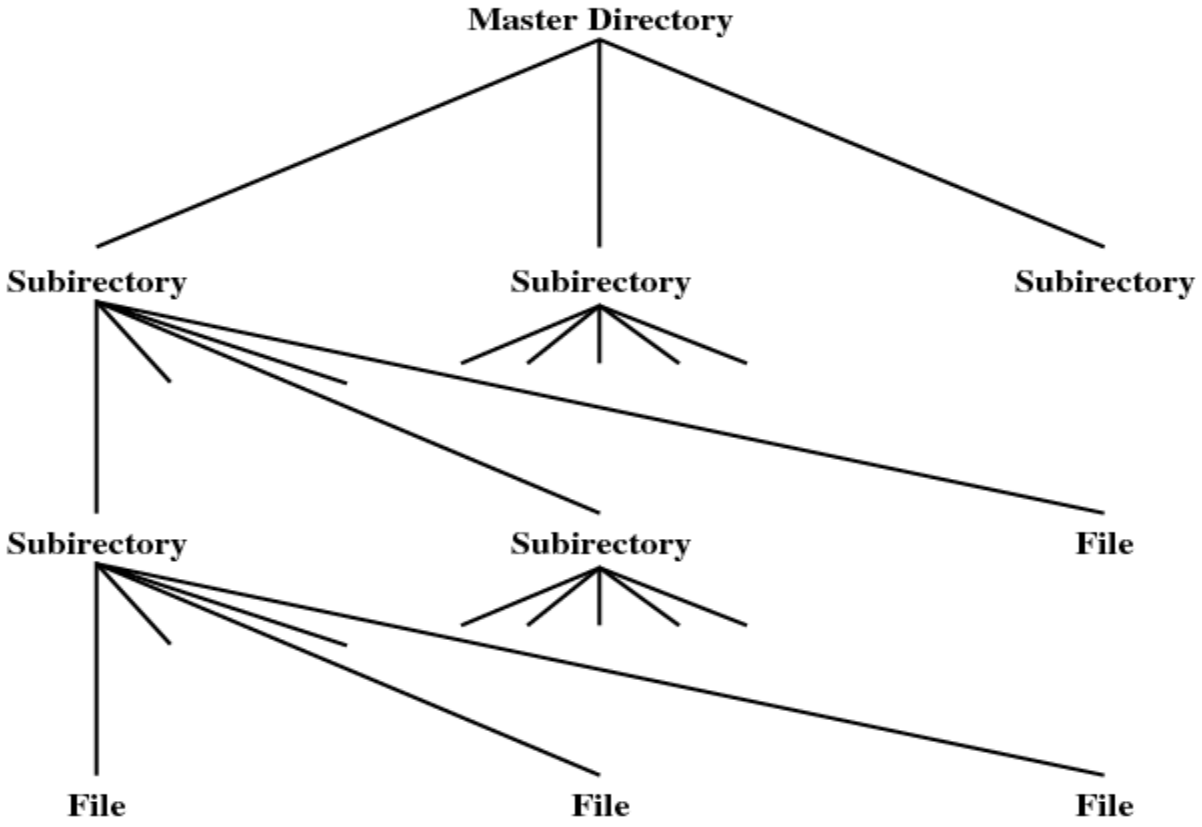


Figure 12.4 Tree-Structured Directory

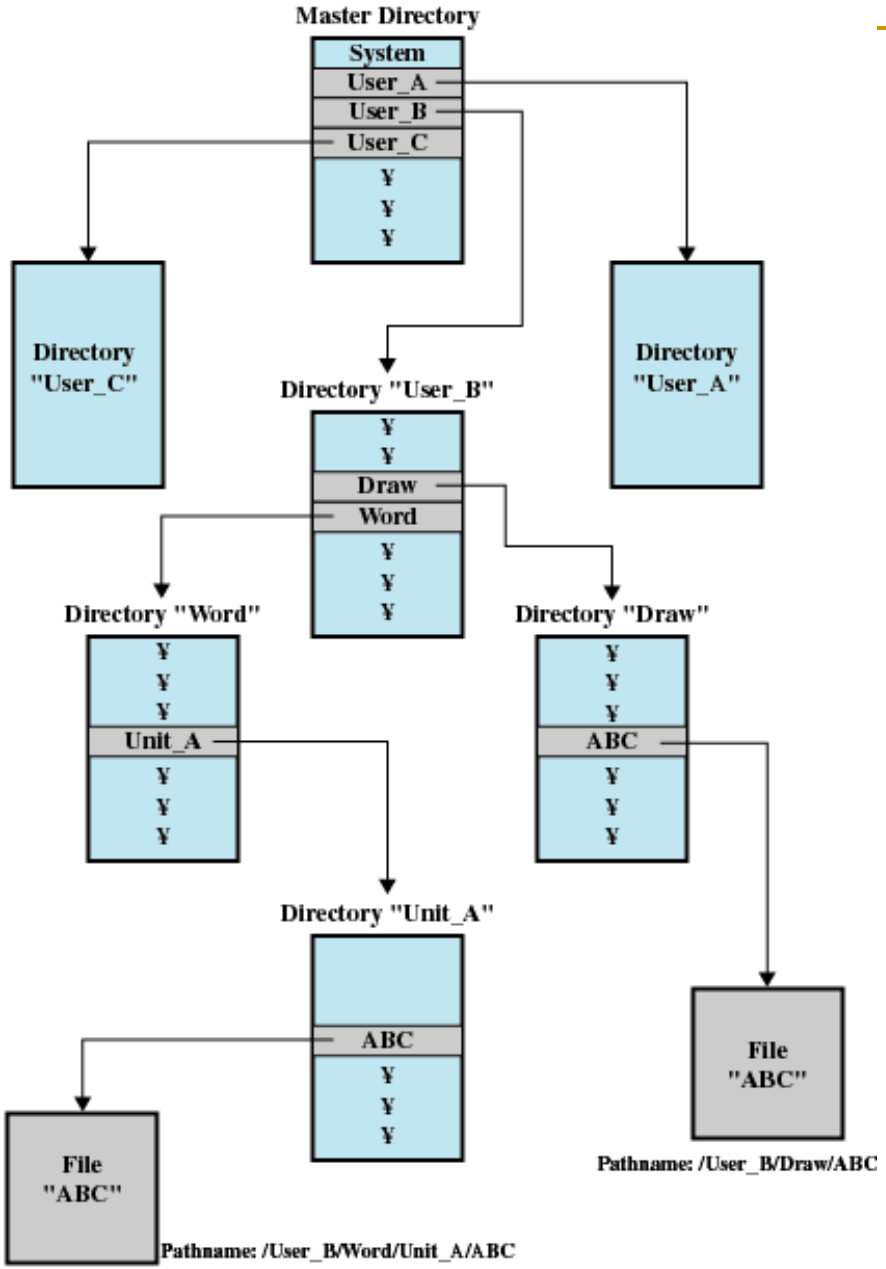


Figure 12.5 Example of Tree-Structured Directory

Contents

- File Management Concept
- **File sharing**
- File organization
- File directories
- Record blocking
- I/O buffering
- Disk Scheduling

File Sharing

- In multiuser system, allow files to be shared among users
- Two issues
  - Access rights
  - Management of simultaneous access

Access Rights

- None
  - User may not know of the existence of the file
  - User is not allowed to read the user directory that includes the file
- Knowledge
  - User can only determine that the file exists and who its owner is

Access Rights

- Execution
  - The user can load and execute a program but cannot copy it
- Reading
  - The user can read the file for any purpose, including copying and execution
- Appending
  - The user can add data to the file at the end but cannot modify or delete any of the file’s contents

Access Rights

- Updating
  - The user can modify, deleted, and add to the file's data. This includes creating the file, rewriting it, and removing all or part of the data
- Changing protection
  - User can change access rights granted to other users
- Deletion
  - User can delete the file

Access Rights

- Owners
  - Has all rights previously listed
  - May grant rights to others using the following classes of users
    - **Specific user**:Individual user
    - **User groups**:a set of users
    - **All** : All users for public files

Simultaneous Access

- User may lock entire file when it is to be updated
- User may lock the individual records during the update
- Mutual exclusion and deadlock are issues for shared access

Contents

- File Management Concept
- File sharing
- File organization
- File directories
- **Record blocking**
- i/O buffering
- Disk Scheduling

Record Blocking

- Records are the logical unit of access of a structured file
- But blocks are the unit for I/O with secondary storage
- To perform I/O records must be organized as blocks.
- Issues related with block
  - Should blocks be fixed or variable length?
    - Most system supports fixed size blocks which simplifies I/O, buffer allocation in main memory and organization of blocks on secondary storage device.
  - What should be the relative size of block be compared to the average record size.

Record Blocking

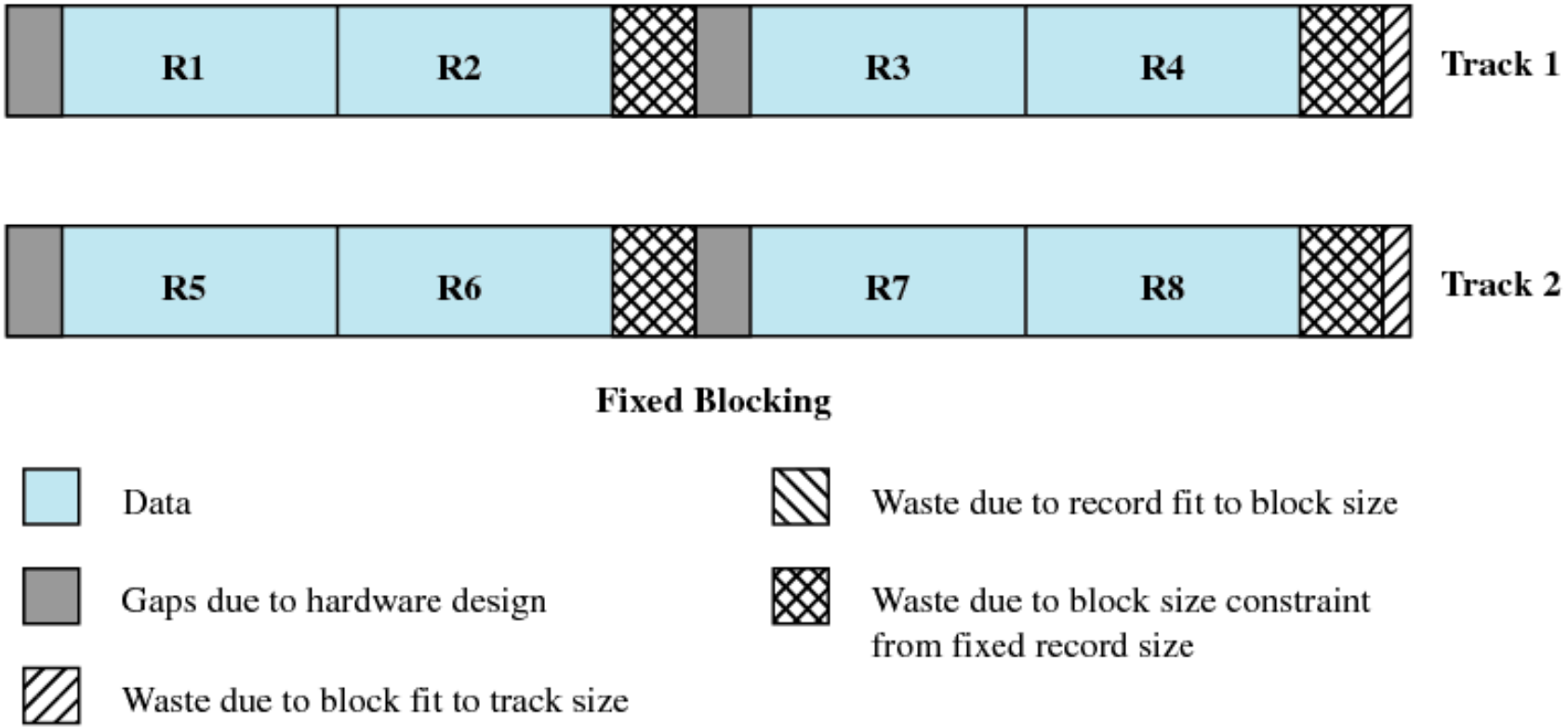
- Given the size of block ,there are Three methods of record blocking

1. Fixed Blocking
2. Variable Blocking: Spanned
3. Variable Blocking Unspanned

Fixed Blocking

- Fixed-length records are used, and an integral number of records are stored in a block.
- Unused space at the end of a block is ***internal fragmentation***
- It is common method for sequential files with fixed length records

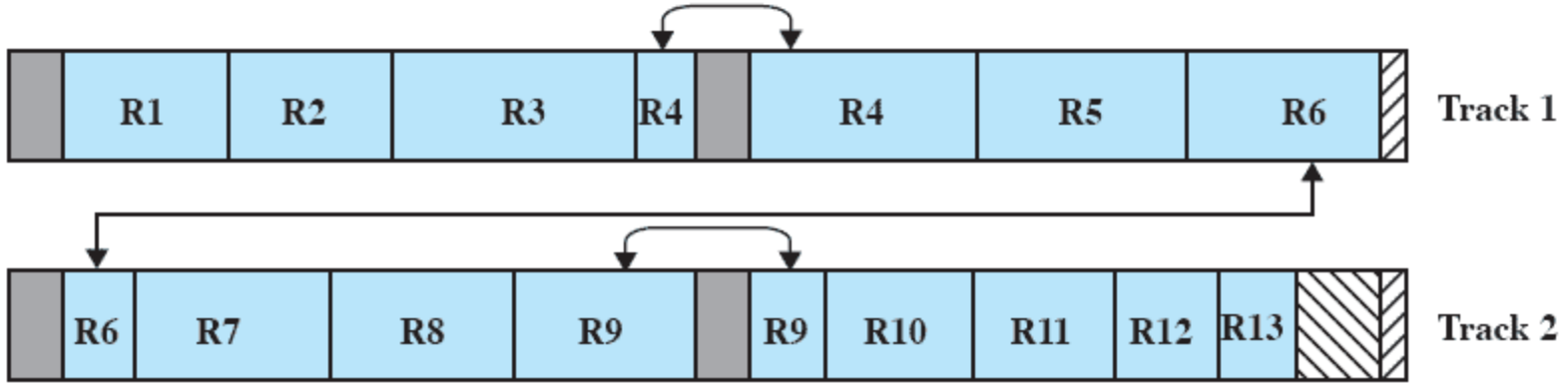
Fixed Blocking



Variable Length ,Spanned Blocking

- Variable-length records are used and are packed into blocks with no unused space.
- Some records may span multiple blocks
  - Continuation is indicated by a pointer to the successor block
  - It is efficient for storage and does not limit the size of records.
  - It is difficult to implement .
  - as record span two blocks require two I/O & files are difficult to update.

Variable Length ,Spanned Blocking



Variable Blocking: Spanned

Data

Gaps due to hardware design

Waste due to block fit to track size

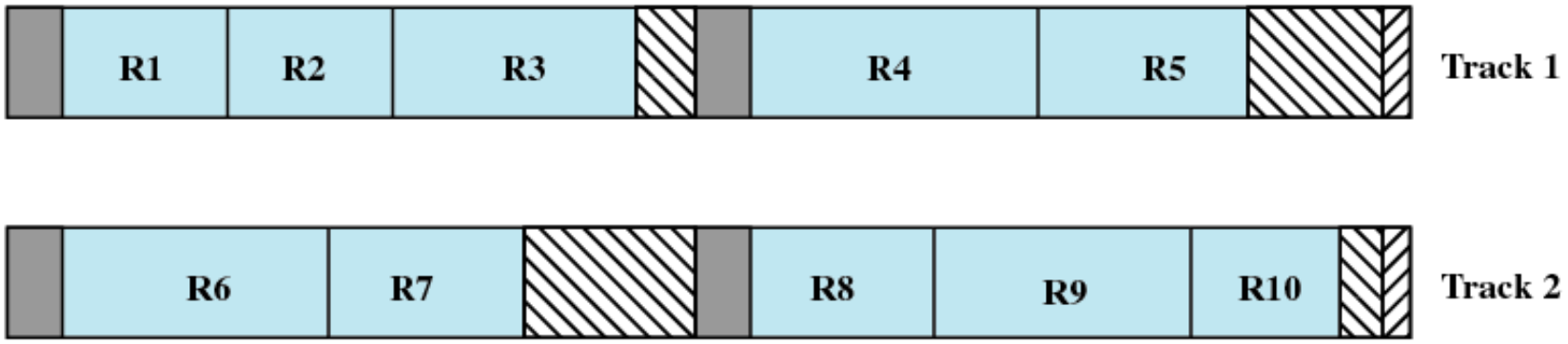
Waste due to record fit to block size

Waste due to block size constraint from fixed record size

Variable Length ,Unspanned Blocking

- Uses variable length records without spanning
- Wasted space in most blocks because of the inability to use the remainder of a block if the next record is larger than the remaining unused space.
- Results in wasted space & limits record size to the size of block.

Variable Blocking Unspanned



Variable Blocking: Unspanned

Data

Gaps due to hardware design

Waste due to block fit to track size

Waste due to record fit to block size

Waste due to block size constraint from fixed record size

Contents

- File Management Concept
- File sharing
- File organization
- File directories
- Record blocking
- **I/O devices**
- Operating System design issues
- I/O buffering
- Disk Scheduling

Introduction

- Humans *interact with machines by providing* information through IO devices.
- Many *on-line services are availed through use of* specialized devices like printers, keyboards etc.
- Management of these devices *can affect the throughput of a system.*

Categories of I/O Devices

- Communication with an IO device is required at the following levels:
  - The need for a *human to input information and* output from a computer.
  - The need for a *device to input information and* receive output from a computer.
  - The need for *computers to communicate over networks.*

Categories of I/O Devices

- Human readable/Interfacing
  - Used to communicate with the user
  - Ex:
    - video display terminals
    - keyboard
    - mouse
    - printer

Categories of I/O Devices

- Machine readable/storage devices
  - Used to communicate with electronic equipment
  - Ex:
    - Disk drives
    - Tape drives
    - Controllers
    - Sensors
    - Actuators
    - USB keys

Categories of I/O Devices

- Communication/transmission
  - Used to communicate with remote devices
  - Ex:
    - digital line drivers
    - Modems
    - controllers

Differences in I/O Devices

- **Data Transfer Rate:** may be difference of several orders of magnitude between the data transfer rates.
- **Application:** the expected use of a device affects the software policies and priorities employed by an operating system
  - disk used to store files must have file-management software
  - disk used to store virtual memory pages needs special hardware to support it
  - terminal used by system administrator or normal user
  - may have different privilege levels

Differences in I/O Devices

- **Complexity of control:** devices such as mice and keyboards require little control (being read-only devices), whereas bidirectional, mirrored disk drives are much more complex.
- **Unit of transfer**
  - data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk
- **Data representation**
  - encoding schemes are used by different devices
- **Error conditions**
  - The error of nature ,the way in which they are reported, their consequences ,available response differs.

I/O Device Data Rates

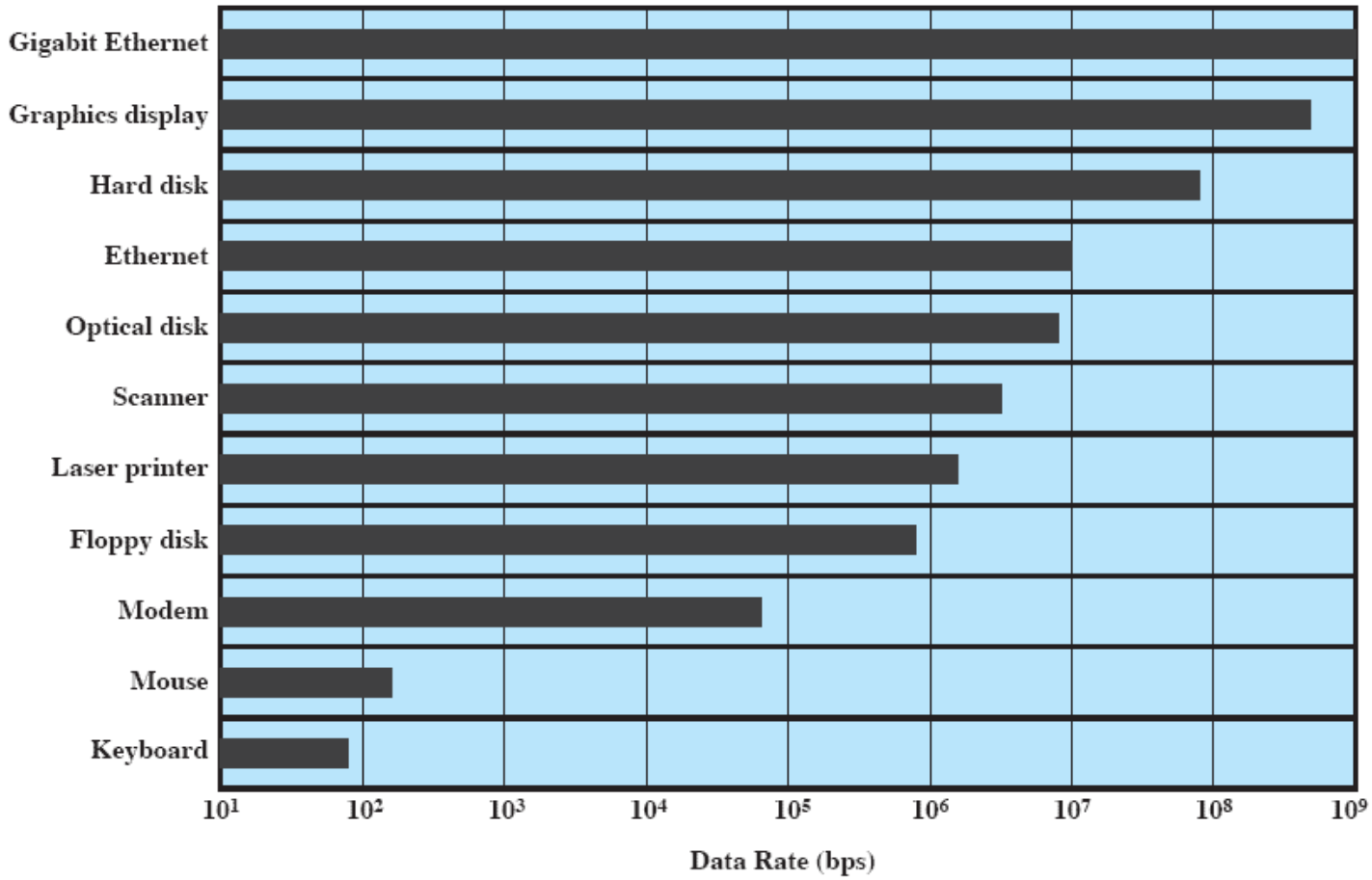


Figure 11.1 Typical I/O Device Data Rates

Contents

- File Management Concept
- File sharing
- File organization
- File directories
- Record blocking
- I/O devices
- **Operating System design issues**
- I/O buffering
- Disk Scheduling

Operating System Design Objectives

- **Efficiency & Generality**
- Efficiency is important as IO operations forms bottleneck in a computing system.
  - I/O extremely slow compared to main memory/processor
  - **Use of multiprogramming** allows for some processes to be waiting on I/O while another process executes.
  - I/O cannot keep up with processor speed
  - **Swapping** is used to bring in additional Ready processes which itself is an I/O operation

Operating System Design Objectives

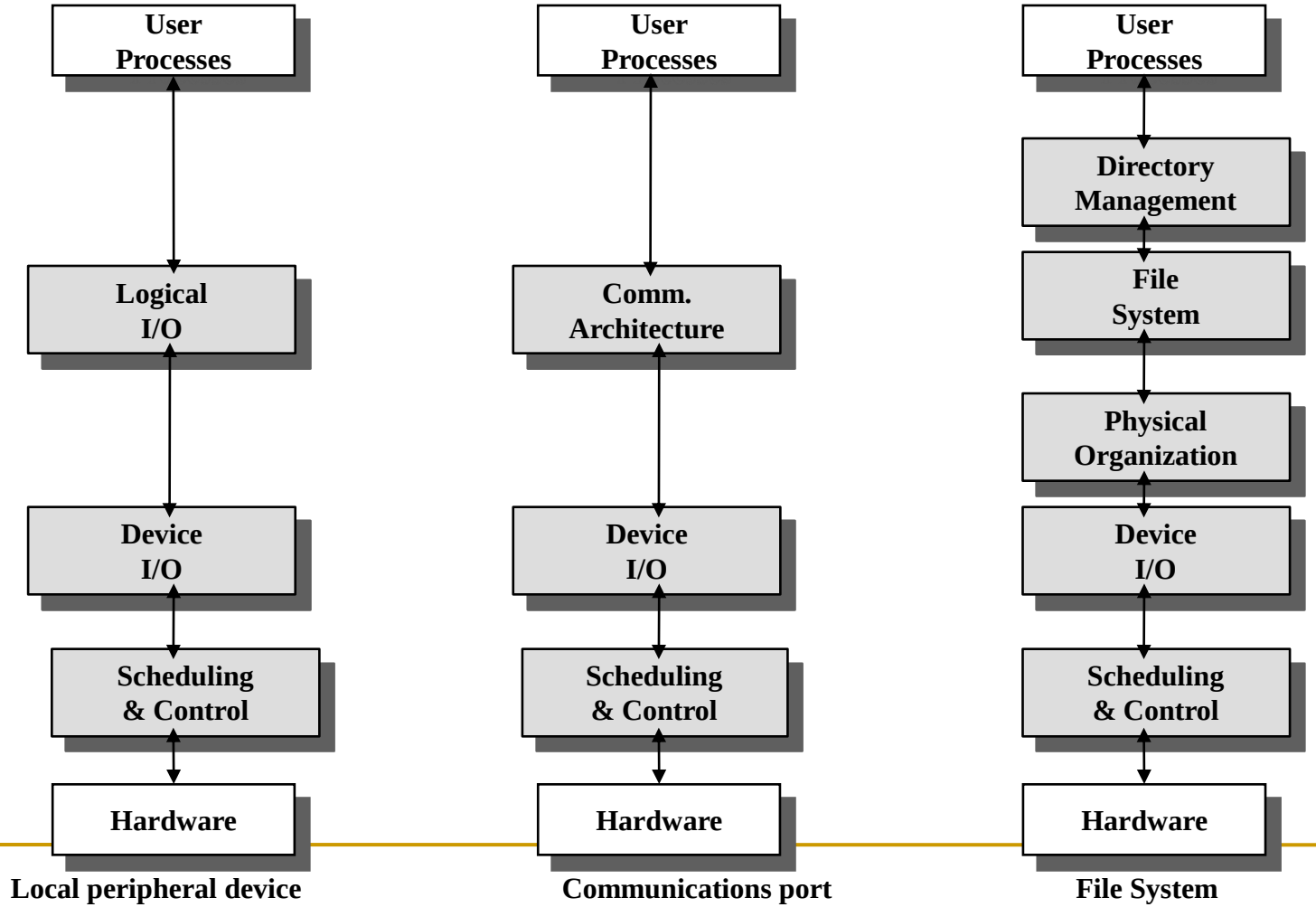
- Generality is an important issue
  - Desirable to handle all I/O devices in a uniform manner.
  - Means the way process view I/O devices and the way by which OS manages I/O devices & operations.
  - How to achieve this?
  - Use hierarchical, modular approach to the design of the I/o function.
  - Hide most of the details of device I/O in lower-level routines so that user processes and upper levels see devices in general terms such as Read, Write, Open, and Close

Hierarchical design

- A hierarchical philosophy leads to organizing an OS into layers.
- Functions of operating system should be separated according to their complexity ,time scale & level of abstraction.
- Each layer performs a related subset of the functions required of the operating system.
- Each layer relies on the next lower layer to perform more primitive functions
- It provides services to the next higher layer.
- Changes in one layer should not require changes in other layers
- At the lowest level of the I/O hierarchy are sections of code which must interact directly with hardware, and complete their activities in a few billionths of a second.

A Model of I/O Organization

Based on Type of device & application



Contents

- File Management Concept
- File sharing
- File organization
- File directories
- Record blocking
- **I/O buffering**
- Disk Scheduling

I/O Buffering

- Scenario: a user process wants to read blocks of data from tape/disk, one at a time
  - Each block = 512 bytes
  - To be stored into virtual address 1000 to 1511
- Simplest way?
  - Execute I/O instruction and wait for data to become available
  - Either busy wait, or process suspension on an interrupt.
- Two Problems
  - The process has to wait a long time before the completion of the operation due to the low speed of the tape device
  - I/O interferes with the swapping mechanism in the operation system.
  - 1000-1511 must always be in memory during block transfer
    - Even if OS wants, it cannot swap out a process completely

I/O Buffering

- There is also a risk of single process deadlock!
  - Process issues I/O command
  - Suspended awaiting result
  - Swapped out before I/O began
  - Process is blocked waiting for I/O completion and I/O is blocked waiting for process be to be swapped back in!
- How to avoid?
  - User memory involved in I/O must be locked immediately before I/O request is issued, even though actual I/O operation may take place after some time

I/O Buffering

- There is a speed mismatch between I/O devices and CPU. This leads to inefficiency in processes being completed.
- To increase the efficiency, it may be 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.
- A buffer is a memory area that stores data while they are transferred b/n two devices or b/n a device & an application.

I/O Buffering

- Reasons for buffering
  - To cope with a speed mismatch b/n the producer & consumer of a data stream.

    Ex: File being received via modem for storage on the hard disk.
  - Use of buffering is to adapt b/n devices that have different data transfer sizes.

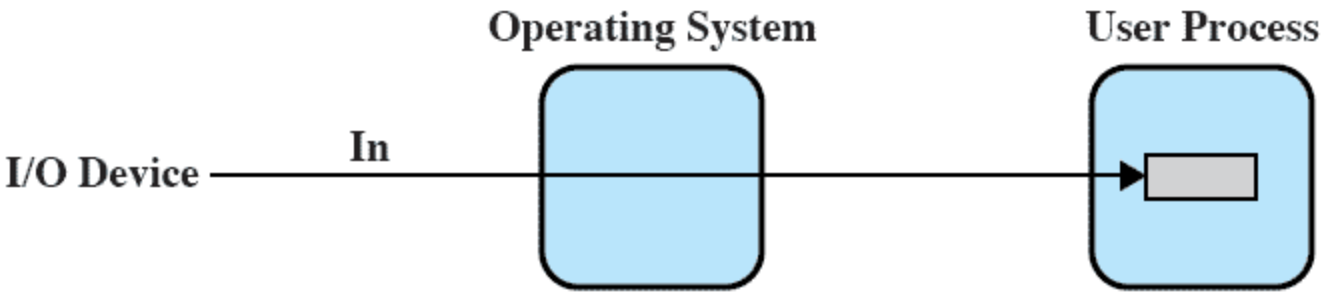
    Ex: in computer networking buffers are used widely for fragmentation & reassembly of msgs.
  - To support copy semantics for application I/O.

    Ex: Example of write () system call

I/O Buffering

- Types of devices
  - Block-oriented
    - information is stored in fixed sized blocks
    - transfers are made a block at a time
    - used for disks and tapes
  - Stream-oriented
    - transfer information as a stream of bytes
    - used for terminals, printers, communication ports, mouse, and most other devices that are not secondary storage

No Buffering

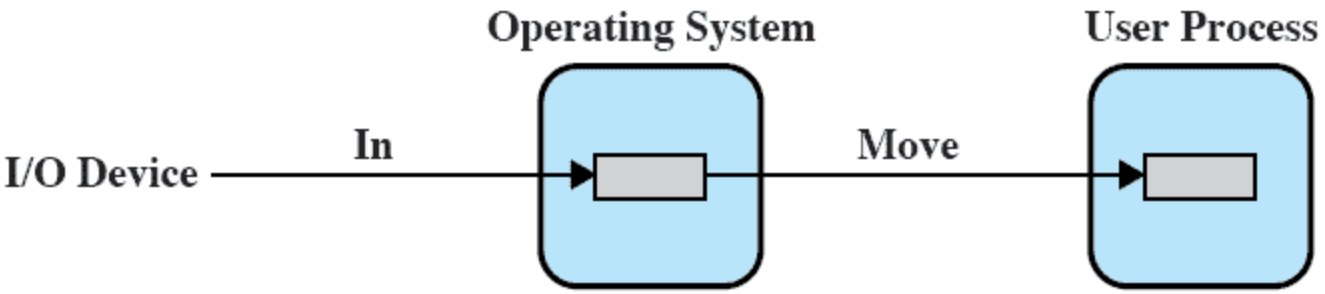


(a) No buffering

Single Buffer (input)

- Operating system assigns a buffer in system portion of the main memory for an I/O request
- Example in case of block-oriented devices
  - Input transfers made to the system buffer
  - When transfer is complete, process moves the block to user space
  - Immediately requests another block
    - Called reading ahead or anticipated input
    - Fair enough assumption except for the “last” block
- User process can now be processing the previous block while the next block is being read
- OS can feel free to swap because I/O is taking place with system memory, not process memory

Single Buffer (input)



(b) Single buffering

Single Buffer

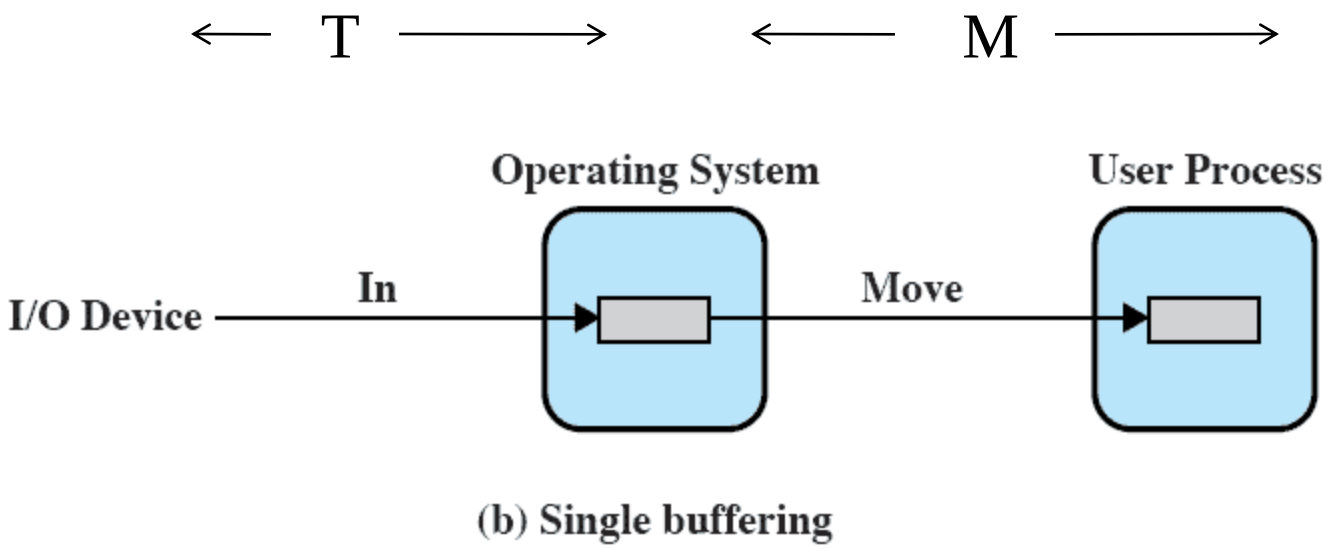
- Block-oriented
  - user process can process one block of data while next block is read in
  - swapping can occur since input is taking place in system memory, not user memory
  - operating system keeps track of assignment of system buffers to user processes
  - output is accomplished by the user process writing a block to the buffer and later actually written out

Single Buffer

- Stream-oriented
  - in a line- at- a time Fashion or a byte-at-a-time fashion.
  - user input from a terminal is one line at a time with carriage return signaling the end of the line
  - output to the terminal is one line at a time
  - For eg. Scroll-mode terminals, line printer
  - In case of byte-at-a-time ,each keystroke is important in sensor,controller.
  - OS & user process follows producer-consumer model

Single Buffer (input)

C:Comutation Time b/w two requests



(b) Single buffering

Without buffering: C+T

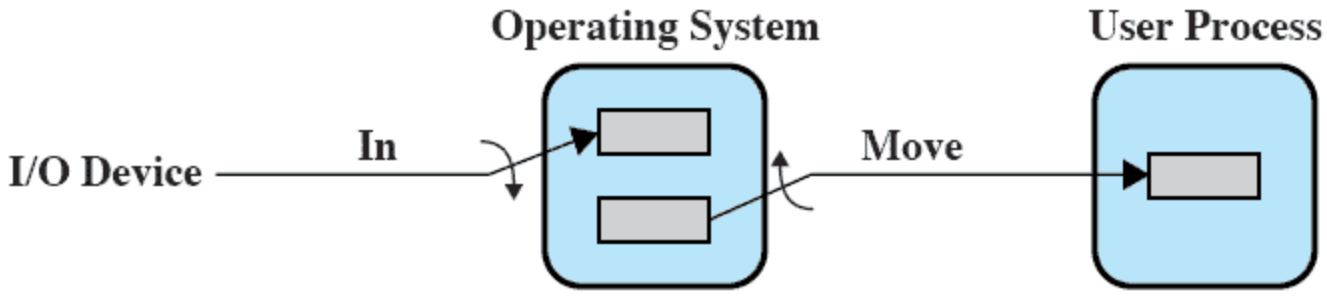
With buffering: max[C,T] + M

Single Buffer

- Suppose that T is required to input one block
- **C** is the computation time that intervenes b/n input requests.
- Without buffering, the execution time per block is essentially **T+C**.
- With a single buffer, the time is max**[C, T] +M**, Where M is the time required to move the data from the system buffer to user memory.
- In most cases execution time per block is less with a single buffer compared to no buffer.

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.
- known as **buffer swapping.**



(c) Double buffering

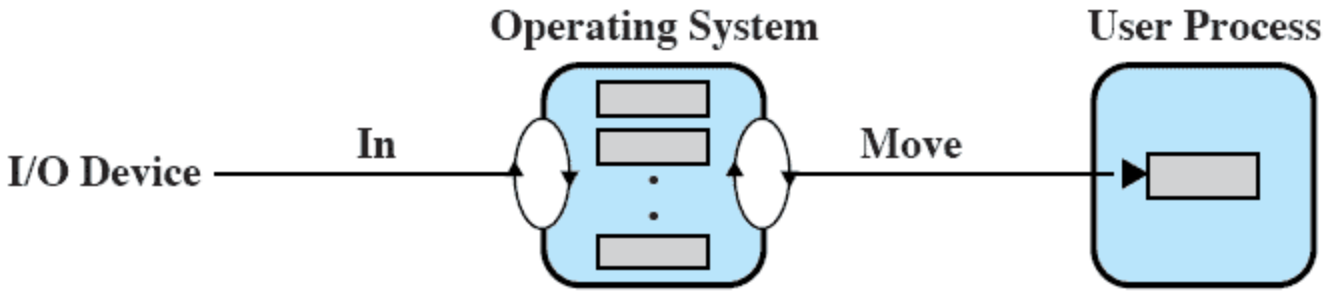
Double Buffer

- For block-oriented transfer, we can roughly estimate the execution time as max[C,T].
- In both cases(C<= T and C>T) an improvement over single buffering is achieved. Again, this improvement comes at the cost of increased complexity.
- For stream-oriented input, there are two alternative modes of operation.
- For line-at-a time I/O, the user process need not be suspended for input or output, unless the process runs ahead of the double buffers.
- For byte-at-a time operation, the double buffer offers no particular advantage over a single buffer of twice the length.

- OS & user process follows producer-consumer model

Circular Buffer

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



(d) Circular buffering

Contents

- File Management Concept
- File sharing
- File organization
- File directories
- Record blocking
- I/O devices
- Operating System design issues
- I/O buffering
- **Disk Scheduling**

Disk Scheduling

Physical organization of data on disk

Track, sector

Seek time

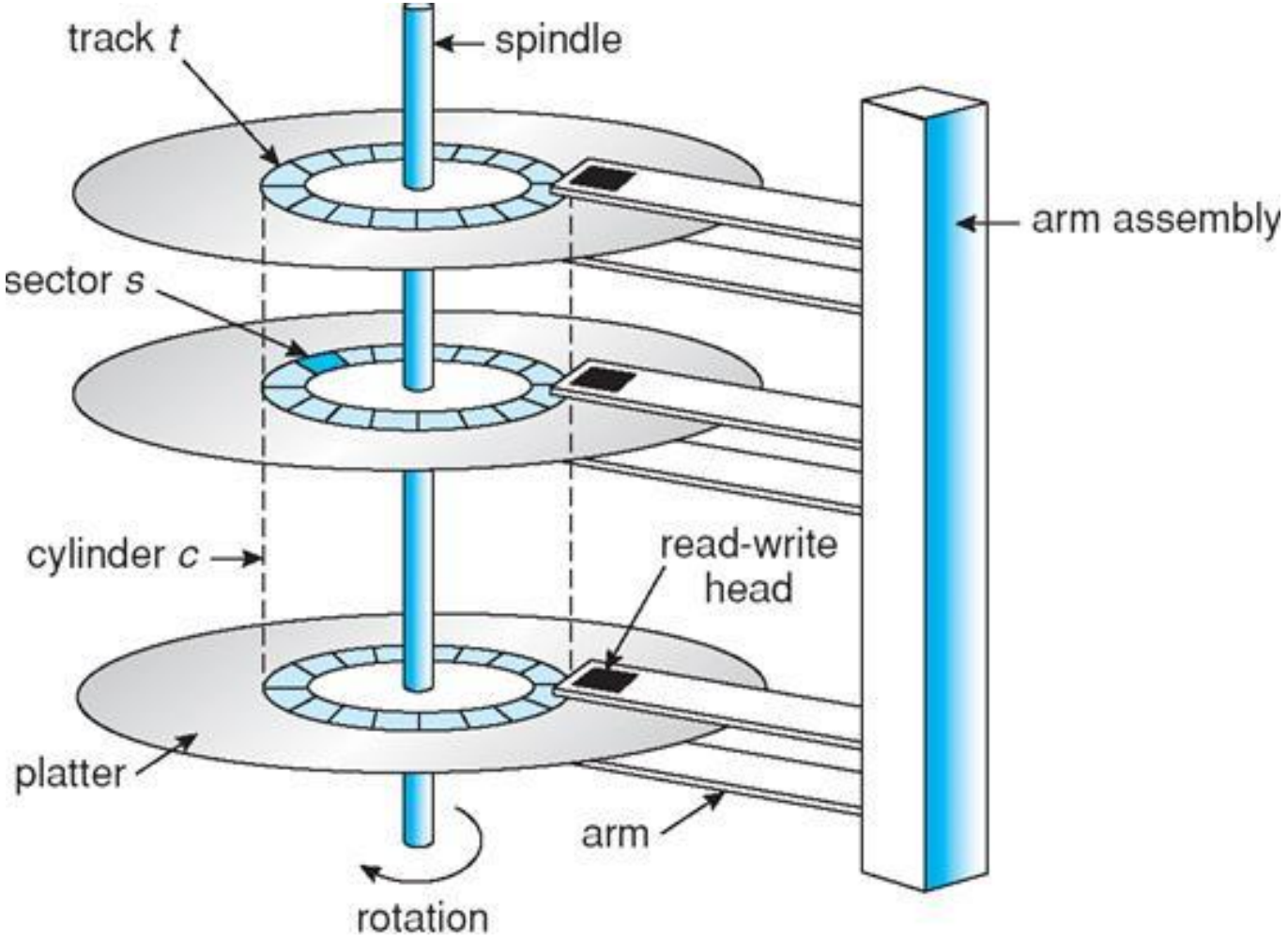
Rotational delay or rotational latency

Access Time

Importance of disk scheduling

Two cases of total time to read a file!!

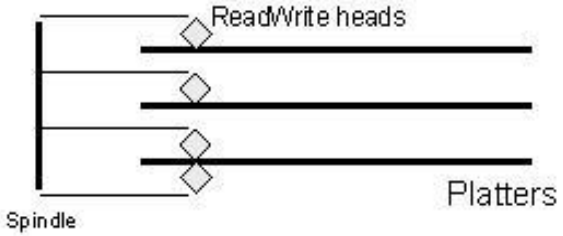
Physical Disk Structure

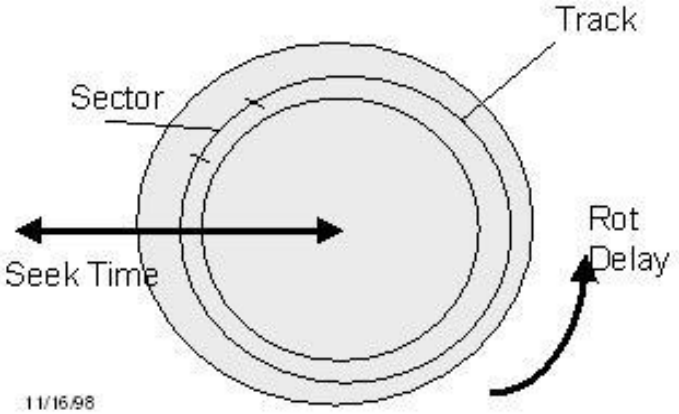


Physical Disk Structure

Physical Disk Structure

- **Disk are made of metal platters with a read/write head flying over it.**
- **To read from disk, we must specify:**
  - cylinder #
  - surface #
  - sector #
  - transfer size
  - memory address
- **Transfer time includes: seek, latency, and transfer time**





Disk Scheduling

Disk Performance Parameters:

- To read or write, the disk head must be positioned at the desired track and at the beginning of the desired sector
- Seek time
  - time it takes to position the head at the desired track
- Rotational delay or rotational latency
  - time its takes until desired sector is rotated to line up with the head

Disk Scheduling

Disk Performance Parameters:

- Access time
  - sum of seek time and rotational delay
  - the time it takes to get in position to read or write
- Data transfer occurs as the sector moves under the head
- Data transfer for an entire file is faster when the file is stored in the same cylinder and in adjacent sectors

Disk Scheduling Policies

- Seek time is the reason for differences in performance
- For a single disk there will be a number of I/O requests
- If requests are selected randomly, we will get the worst possible performance

- FCFS 3/FRAMES :15
- 4 FRAMS:9
- LRU: 3 /F :12
- 4/F:7
- OPTIMAL 3/F :9
- 4/F :8

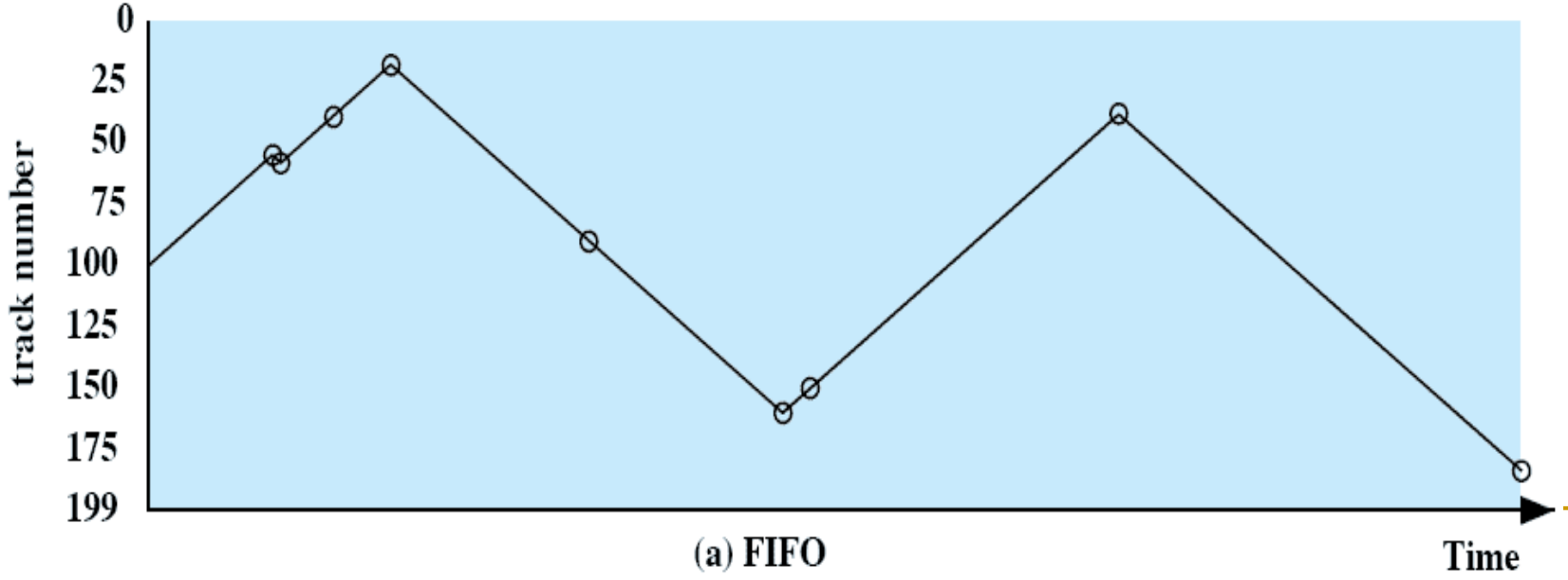
Disk Scheduling Example

we assume that the disk head is initially located at track 100. In this example, we assume a disk with 200 tracks and that the disk request queue has random requests in it. 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 (FIFO)

uler, are 55, 58, 39, 18, 90, 160, 150, 38, 184.

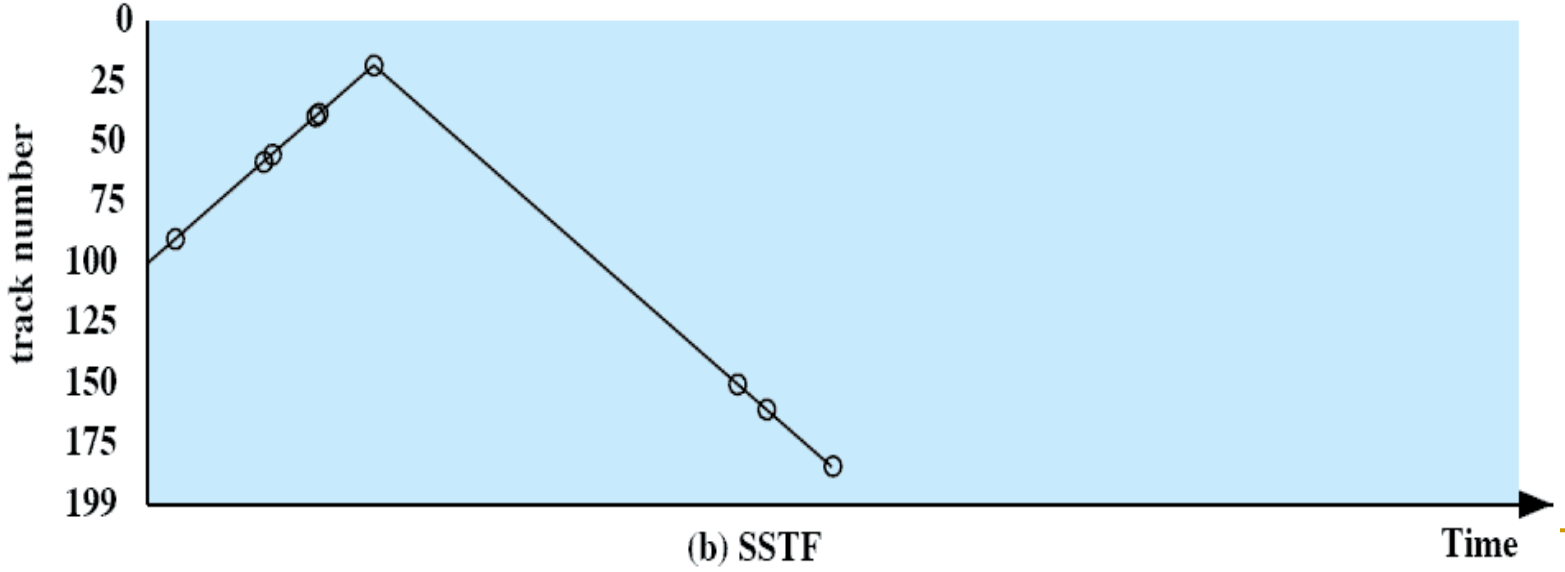
- Process request sequentially
- Fair to all processes
- Approaches random scheduling in performance if there are many processes



Shortest Service Time First (SSTF)

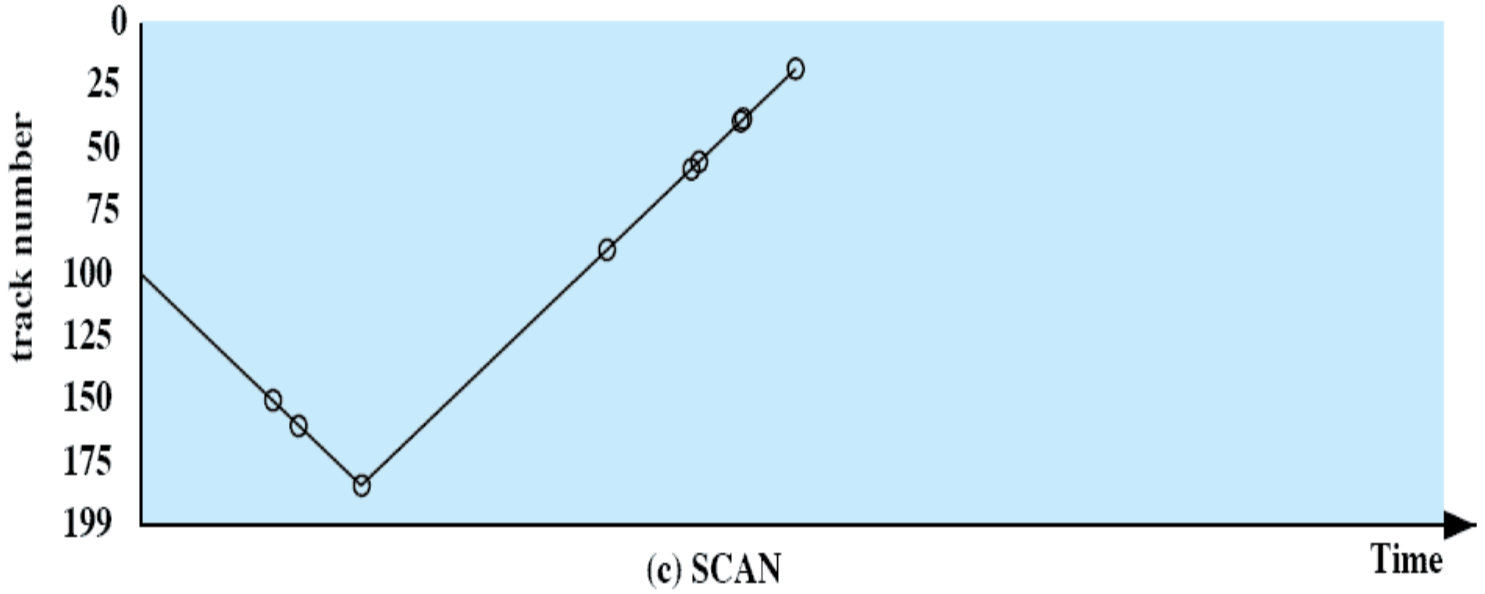
uler, are 55, 58, 39, 18, 90, 160, 150, 38, 184.

- 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



SCAN

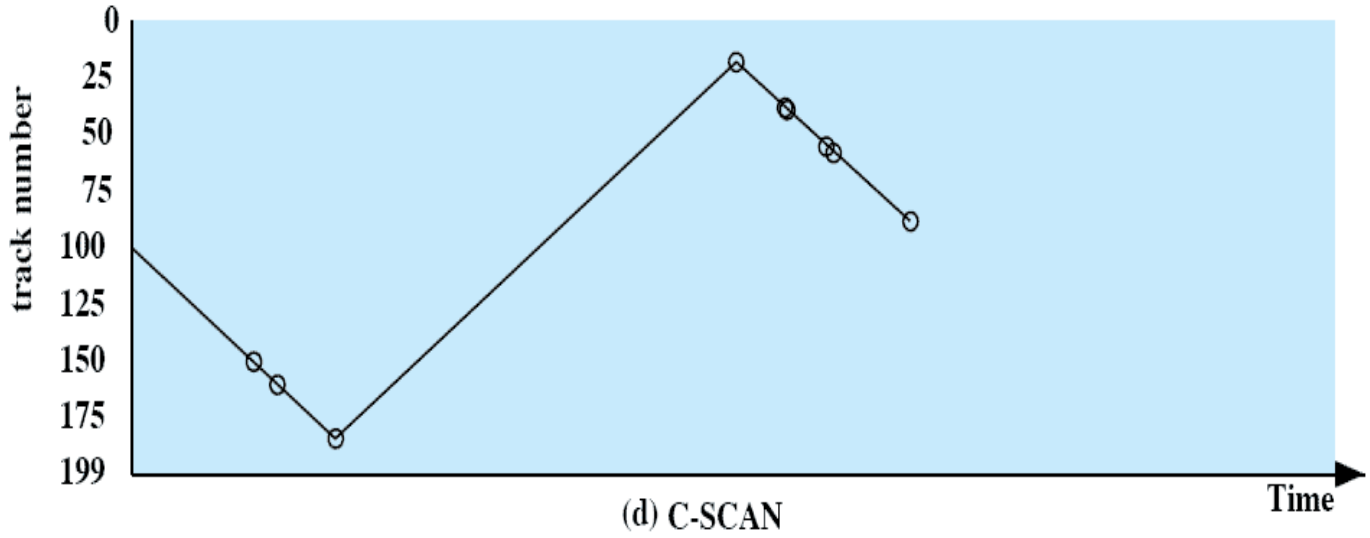
- Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction then the direction is reversed



(c) SCAN

C-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

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

Selecting a Disk-Scheduling Algorithm

- SSTF is common
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or SCAN is a reasonable choice for the default algorithm.

- Suppose the order of request is-

  (82,170,43,140,24,16,190)

  And current position of Read/Write head is : 50
- **87, 160, 40, 140, 36, 72, 66, 15**

  In that order, if the disk head is initially at cylinder 60.

FCFS

- (60 - 87)+(87 - 160)+(160 - 40)+(40 - 140)+(140 - 36)+(36 - 72)+(72 - 66)+(66 - 15)

  = (27+73+120+100+104+36+6+51)

  = 517 cylinders

  ∴ The average head movements are=

  517/8 = 64.625 cylinders

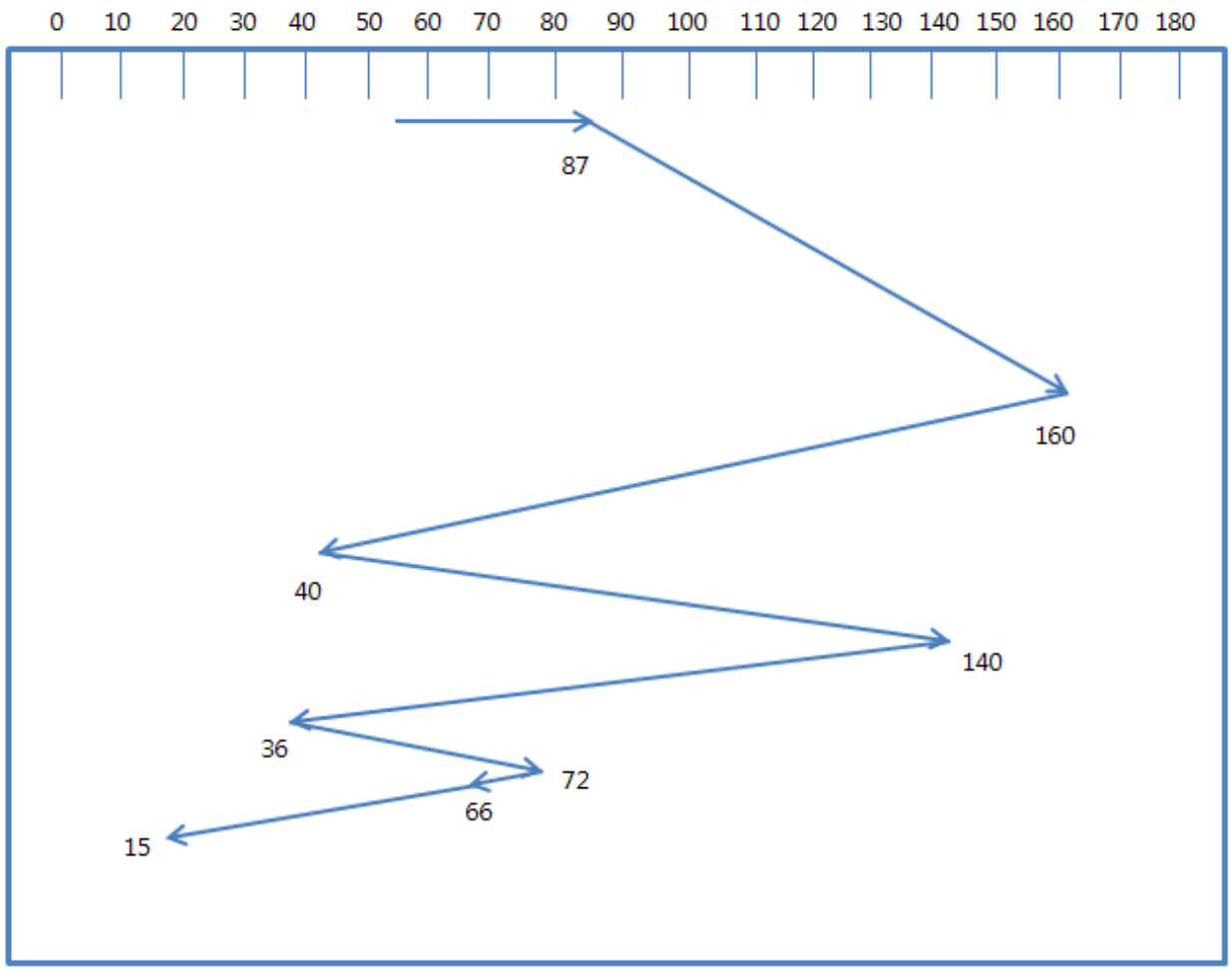


Fig: FCFS Scheduling

SSTF

- The total head movements of this algorithm is

  =(60 - 66)+(66 - 72)+(72 - 87)+(87 - 40)+(40 - 36)+(36 - 15)+(15 - 140)+(140 - 160)

  = (6+6+15+47+4+21+125+20)

  = 244 cylinders

  ∴ The average head movements are = 244/8 = 30.s5 cylinders

SCAN

- The total head movements of this algorithm is

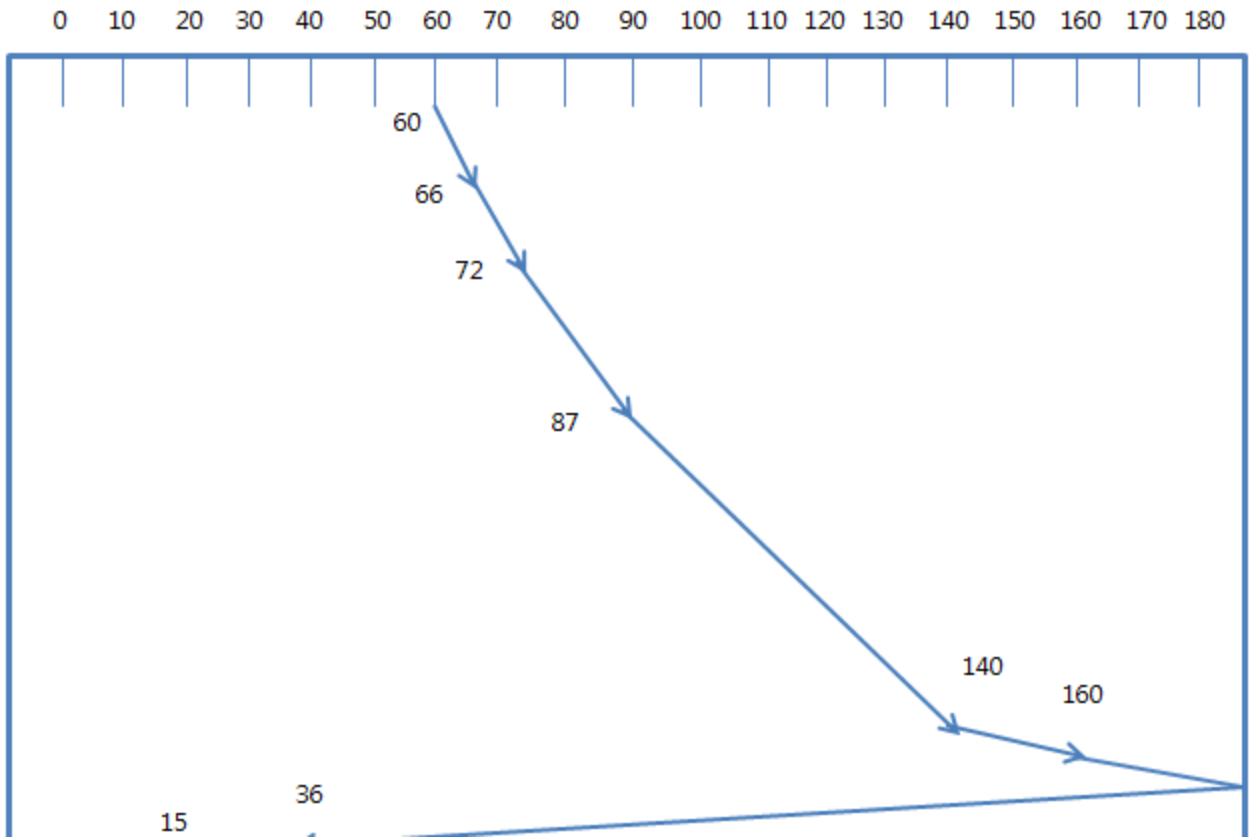
  =(60 - 66)+(66 - 72)+(72 - 87)+(87 - 140)+(140 - 160)+(160-199) 199-15

  = (6+6+15+53+20+39)

- = 323 cylinders

  ∴ The average head movements are = 323/8 = 40.375 cylinders

SCAN


0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180
60
66
72
87
140
160
36
15

C SCAN

- 3

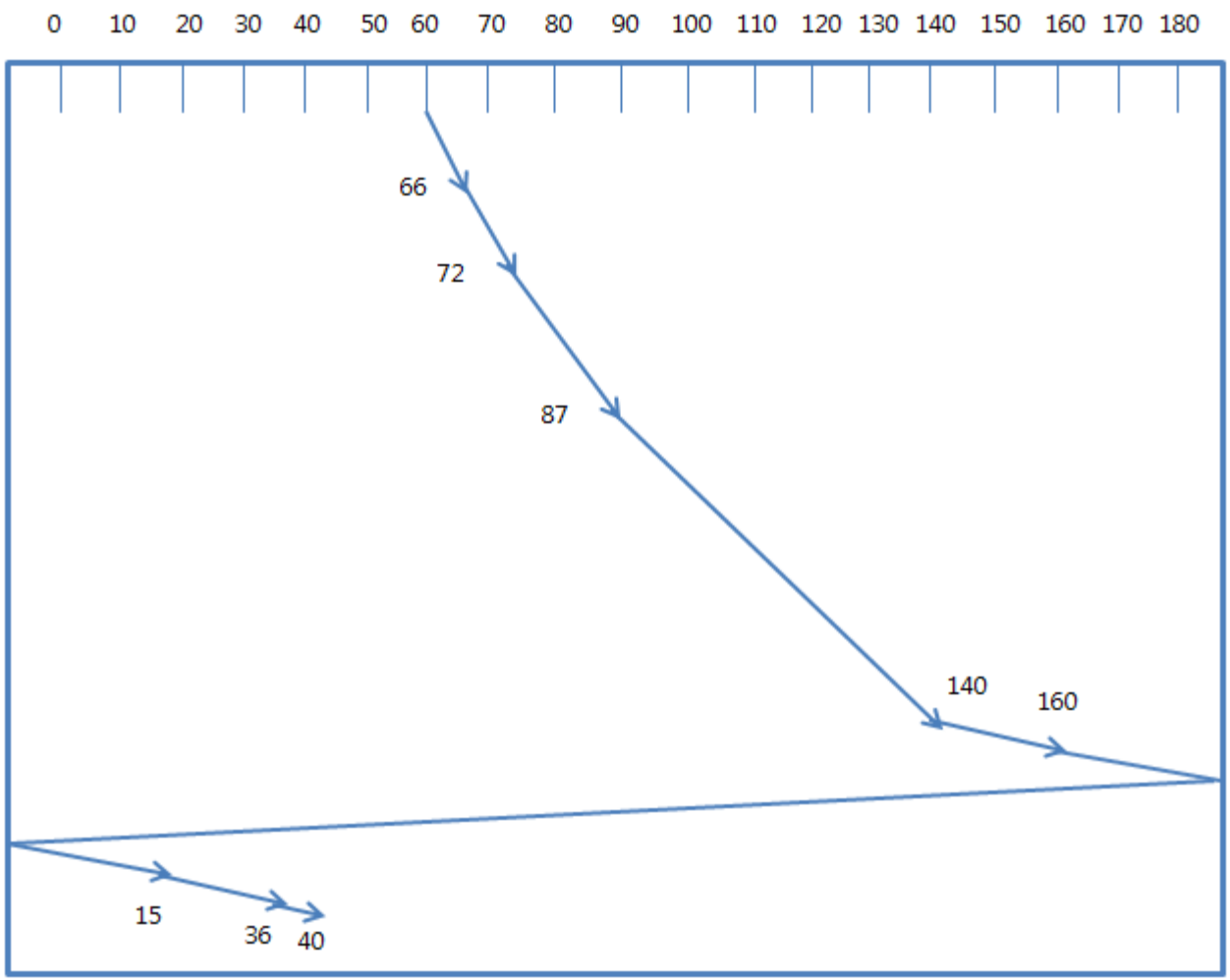


Fig: C-SCAN Scheduling

C LOOK

- C look 2
- Look 24

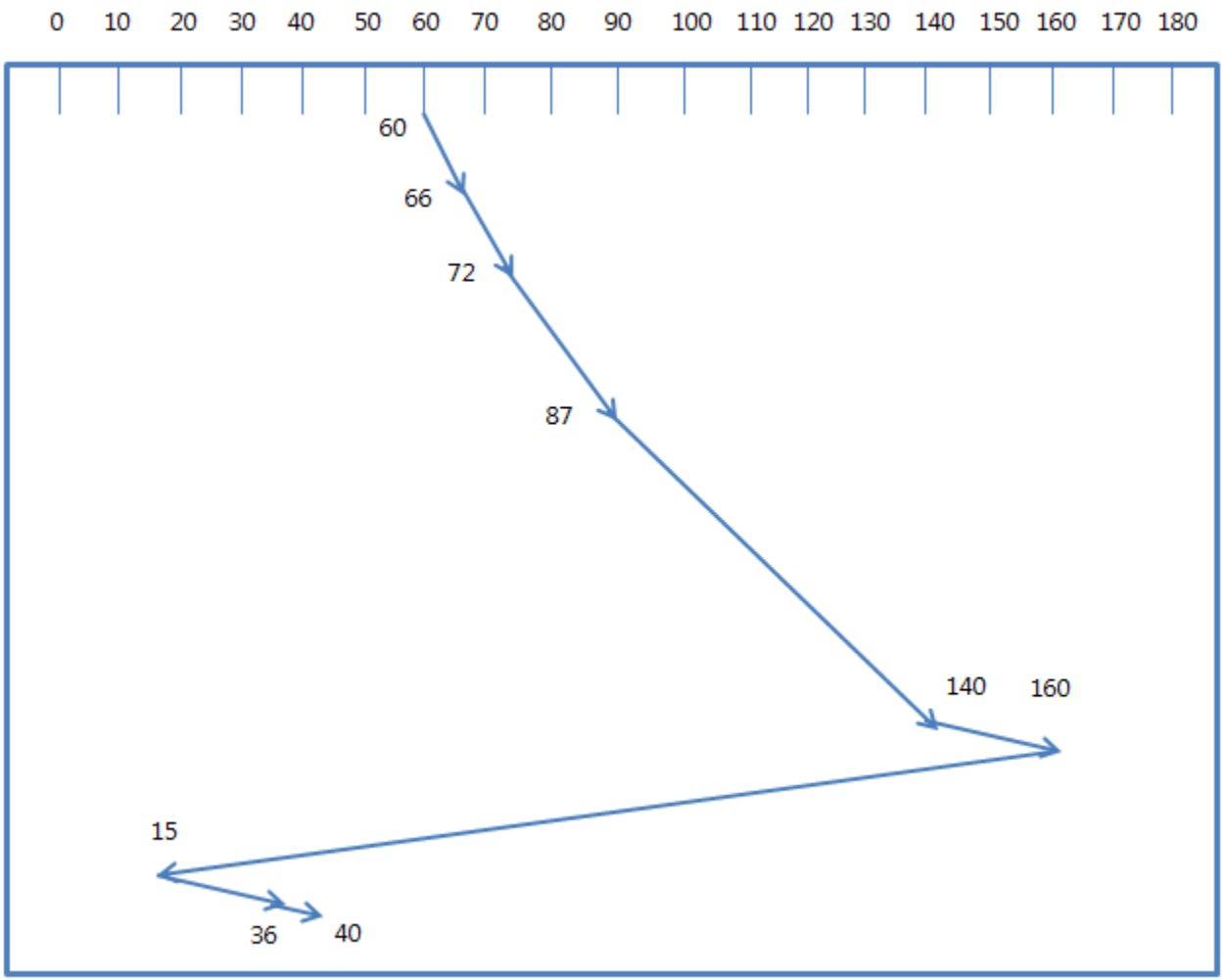


Fig: Look Scheduling