20CST43 - OPERATING SYSTEMS

UNIT V

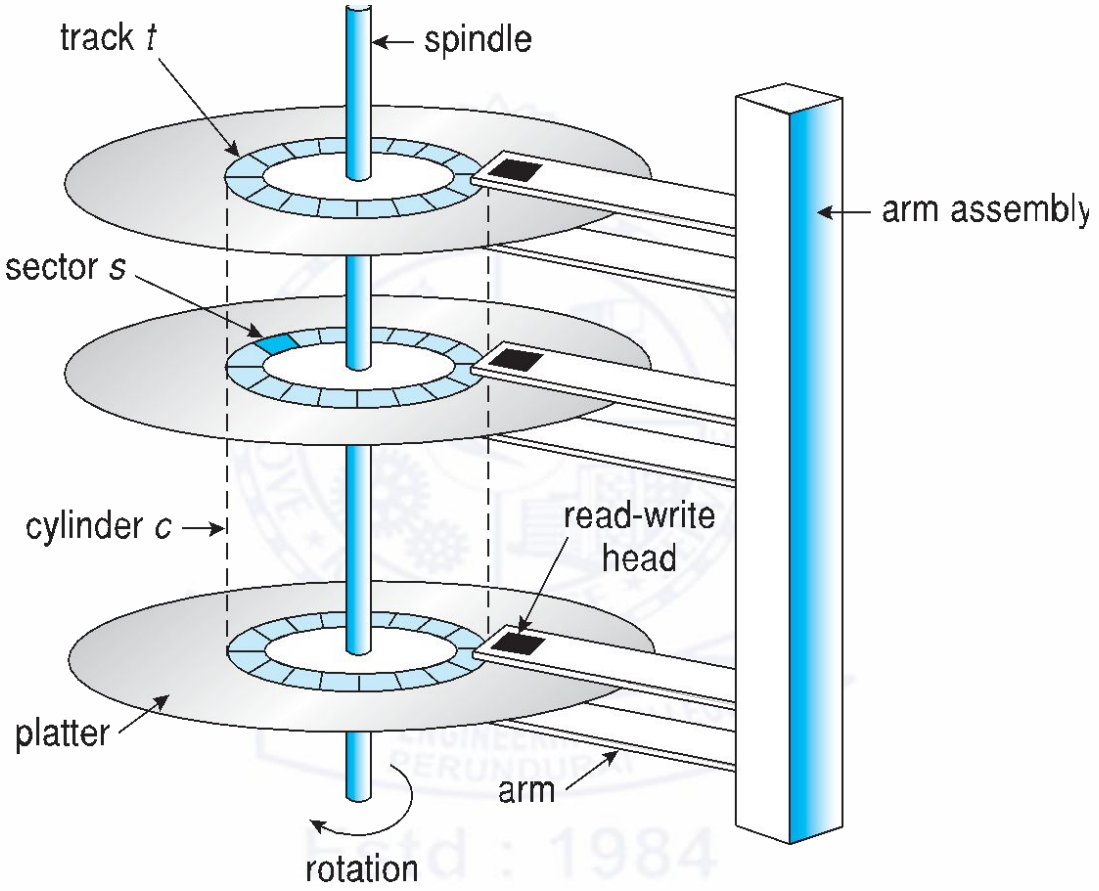
UNIT V

- **Storage Management:** Mass Storage Structure: Overview – HDD Scheduling. File System: File Concept – Access Methods – Directory Structure – Protection. File System Implementation: File System Structure – File System Operations – Directory Implementation – Allocation Methods - Free Space Management. – Security : The Security Problem – program Threats - Case study: Linux System.

Overview of Mass Storage Structure

- Bulk of secondary storage for modern computers is **hard disk drives** (**HDDs**) and **nonvolatile memory** (**NVM**) devices
- **HDDs** spin platters of magnetically-coated material under moving read-write heads
  - Drives rotate at 60 to 250 times per second
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface -- That's bad
- Disks can be removable

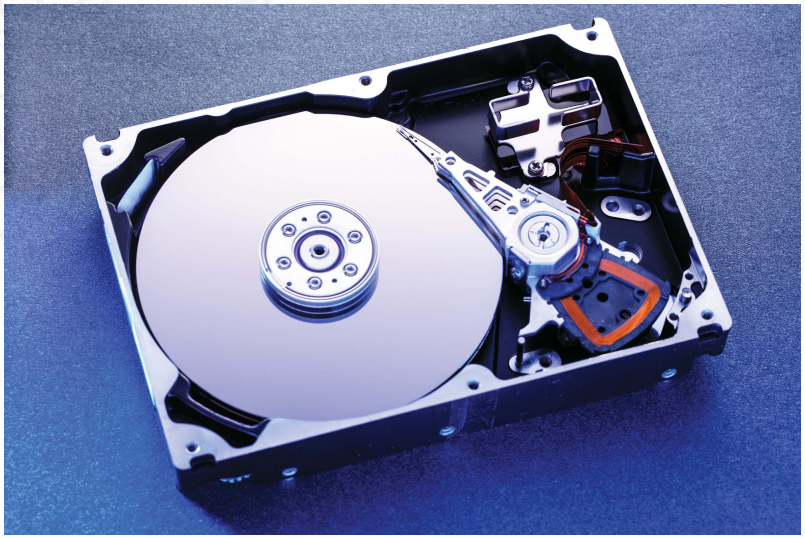
Moving-head Disk Mechanism



https://animagraffs.com/hard-disk-drive/

Hard Disk Drives

- Platters range from .85” to 14” (historically)
  - Commonly 3.5”, 2.5”, and 1.8”
- Range from 30GB to 3TB per drive
- Performance
  - Transfer Rate – theoretical – 6 Gb/sec
  - Effective Transfer Rate – real – 1Gb/sec
  - Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Latency based on spindle speed



Hard Disk Performance

- **Access Latency** = **Average access time** = average seek time + average latency
  - For fastest disk 3ms + 2ms = 5ms
  - For slow disk 9ms + 5.56ms = 14.56ms
- Average I/O time = average access time + (amount to transfer / transfer rate) + controller overhead
- For example to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead =
  - 5ms + 4.17ms + 0.1ms + transfer time =
  - Transfer time = 4KB / 1Gb/s * 8Gb / GB * 1GB / 1024^2KB = 32 / (1024^2) = 0.031 ms
  - Average I/O time for 4KB block = 9.27ms + .031ms = 9.301ms

The First Commercial Disk Drive

1956
IBM RAMDAC computer included the IBM Model 350 disk storage system

5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

Nonvolatile Memory Devices

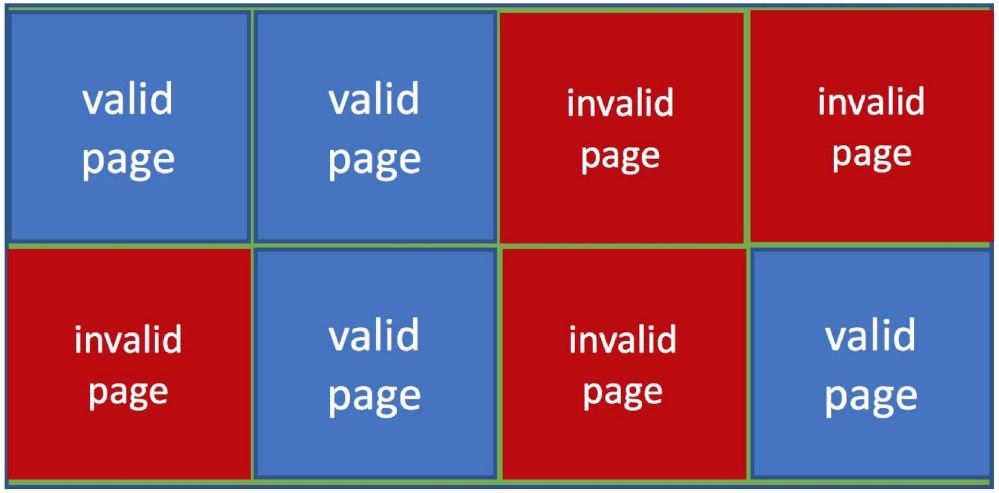
- Flash-memory-based NVM is frequently used in a disk-drive-like container is called a **solid-state disks (SSDs)**
- Other forms include **USB drives** (thumb drive, flash drive), DRAM disk replacements, surface-mounted on motherboards, and main storage in devices like smartphones
- Can be more reliable than HDDs
- More expensive per MB
- May have shorter life span – need careful management
- Less capacity
- But much faster
- No moving parts, so no seek time or rotational latency

Nonvolatile Memory Devices

- Have characteristics that present challenges
- Read and written in “page” increments (think sector) but can’t overwrite in place
  - Must first be erased, and erases happen in larger ”block” increments
  - Can only be erased a limited number of times before worn out – ~ 100,000
  - Life span measured in **drive writes per day** (**DWPD**)
    - A 1TB NAND drive with rating of 5DWPD is expected to have 5TB per day written within warrantee period without failing

NAND Flash Controller Algorithms

- With no overwrite, pages end up with mix of valid and invalid data
- To track which logical blocks are valid, controller maintains **flash translation layer** (**FTL**) table
- Also implements **garbage collection** to free invalid page space
- Allocates **overprovisioning** to provide working space for GC
- Each cell has lifespan, so **wear leveling** needed to write equally to all cells



NAND block with valid and invalid pages

Volatile Memory

- DRAM frequently used as mass-storage device
  - Not technically secondary storage because volatile, but can have file systems, be used like very fast secondary storage
- **RAM drives** (with many names, including RAM disks) present as raw block devices, commonly file system formatted
- Computers have buffering, caching via RAM, so why RAM drives?
  - Caches / buffers allocated / managed by programmer, operating system, hardware
  - RAM drives under user control
  - Found in all major operating systems
    - Linux `/dev/ram`, macOS `diskutil` to create them, Linux `/tmp` of file system type `tmpfs`
- Used as high speed temporary storage
  - Programs could share bulk data, quickly, by reading/writing to RAM drive

Magnetic Tape

Magnetic tape was used as an early secondary-storage medium. Although it is nonvolatile and can hold large quantities of data, its access time is slow compared with that of main memory and drives. In addition, random access to magnetic tape is about a thousand times slower than random access to HDDs and about a hundred thousand times slower than random access to SSDs so tapes are not very useful for secondary storage. Tapes are used mainly for backup, for storage of infrequently used information, and as a medium for transferring information from one system to another.

A tape is kept in a spool and is wound or rewound past a read–write head. Moving to the correct spot on a tape can take minutes, but once positioned, tape drives can read and write data at speeds comparable to HDDs. Tape capacities vary greatly, depending on the particular kind of tape drive, with current capacities exceeding several terabytes. Some tapes have built-in compression that can more than double the effective storage. Tapes and their drivers are usually categorized by width, including 4, 8, and 19 millimeters and 1/4 and 1/2 inch. Some are named according to technology, such as LTO-6 (Figure 11.5) and SDLT.

Figure 11.5 An LTO-6 Tape drive with tape cartridge inserted.

Disk Attachment

- Host-attached storage accessed through I/O ports talking to **I/O busses**
- Several busses available, including **advanced technology attachment** (**ATA**), **serial ATA** (**SATA**), **eSATA**, **serial attached SCSI** (**SAS**), **universal serial bus** (**USB**), and **fibre channel** (**FC**).
- Most common is SATA
- Because NVM much faster than HDD, new fast interface for NVM called **NVM express** (**NVMe**), connecting directly to PCI bus
- Data transfers on a bus carried out by special electronic processors called **controllers** (or **host-bus adapters**, **HBAs**)
  - Host controller on the computer end of the bus, device controller on device end
  - Computer places command on host controller, using memory-mapped I/O ports
    - Host controller sends messages to device controller
    - Data transferred via DMA between device and computer DRAM

Address Mapping

- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
  - Low-level formatting creates **logical blocks** on physical media
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track on the outermost cylinder
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
  - Logical to physical address should be easy
    - Except for bad sectors
    - Non-constant # of sectors per track via constant angular velocity

HDD Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Minimize seek time
- Seek time ≈ seek distance
- Disk **bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

Disk Scheduling (Cont.)

- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
  - Optimization algorithms only make sense when a queue exists
- In the past, operating system responsible for queue management, disk drive head scheduling
  - Now, built into the storage devices, controllers
  - Just provide LBAs, handle sorting of requests
    - Some of the algorithms they use described next

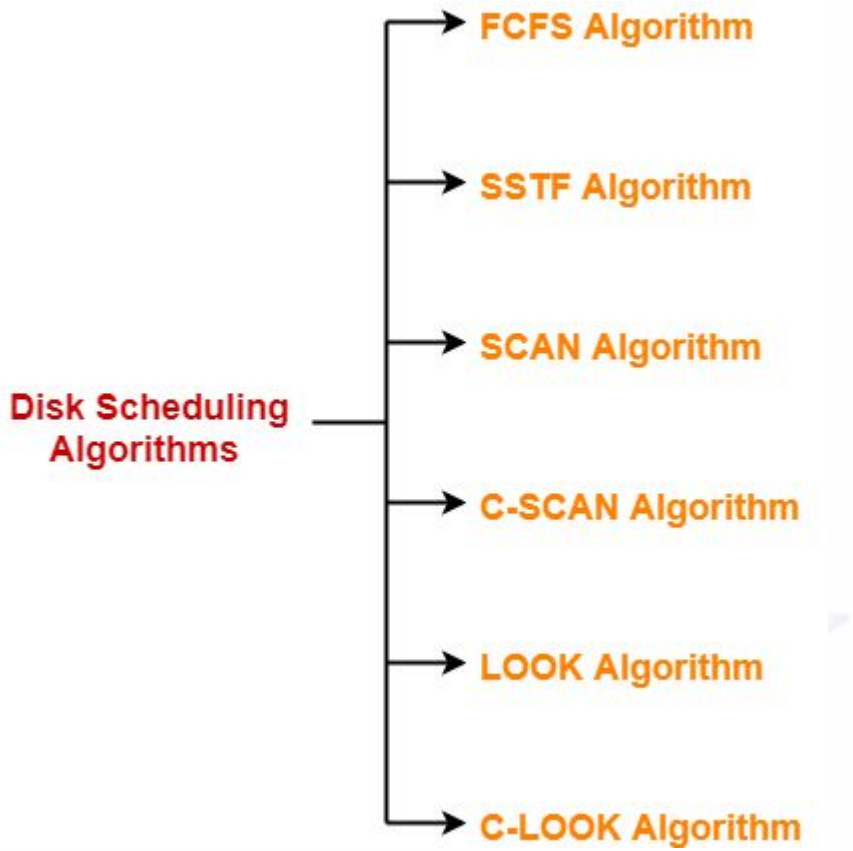
Disk Scheduling (Cont.)

- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying “depth”)
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request queue (0-199)

98, 183, 41, 122, 14, 124, 65, 67

Head pointer 53

Disk Scheduling Algorithms

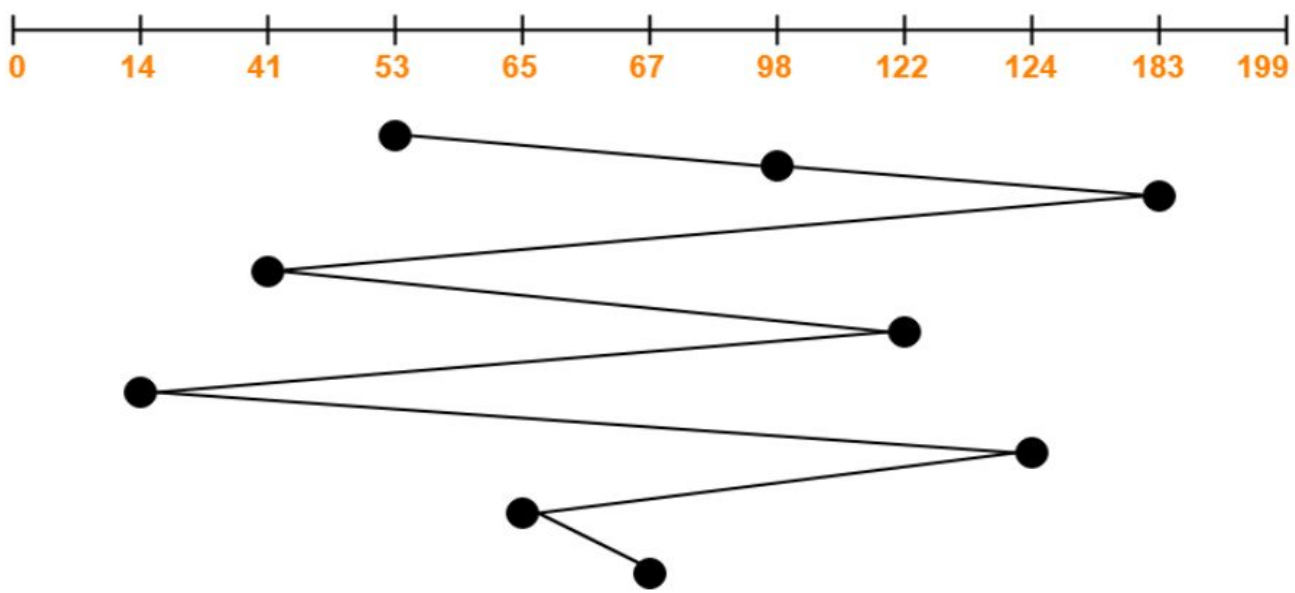


FCFS Scheduling

- Simplest, perform operations in order requested
- no reordering of work queue
- no **starvation**: every request is serviced
- Doesn't provide fastest service
  - **Advantages-**
    - It is simple, easy to understand and implement.
    - It does not cause starvation to any request.
  - **Disadvantages-**
    - It results in increased total seek time.
    - It is inefficient.

Problem

Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The FCFS scheduling algorithm is used. The head is now at cylinder number 53 and previously at 50. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.



Total head movements incurred while servicing these requests

= (98 – 53) + (183 – 98) + (183 – 41) + (122 – 41) + (122 – 14) + (124 – 14) + (124 – 65) + (67 – 65)

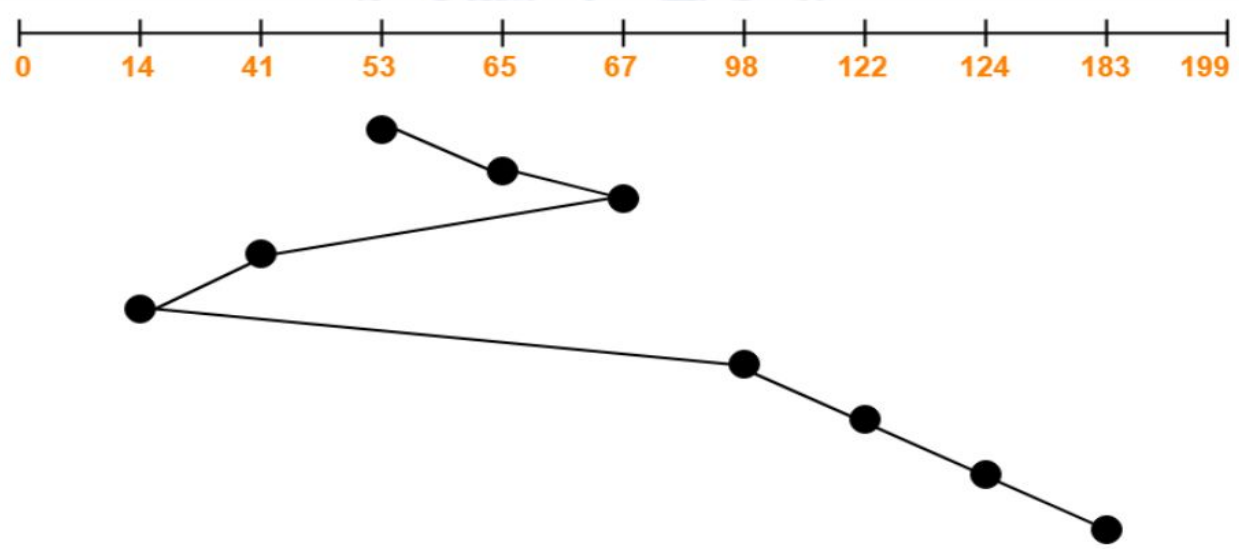
= 45 + 85 + 142 + 81 + 108 + 110 + 59 + 2 = 632

SSTF Scheduling

- SSTF stands for **Shortest Seek Time First**.
- This algorithm services that request next which requires least number of head movements from its current position regardless of the direction.
- It breaks the tie in the direction of head movement.
- **Advantages**
  - It reduces the total seek time as compared to FCFS.
  - It provides increased throughput.
  - It provides less average response time and waiting time.
- **Disadvantages**
  - **starvation** is possible; stay in one area of the disk if very busy
  - switching directions slows things down
  - Not the most optimal

Problem

Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The SSTF scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.



Total head movements incurred while servicing these requests

= (65 – 53) + (67 – 65) + (67 – 41) + (41 – 14) + (98 – 14) + (122 – 98) + (124 – 122) + (183 – 124)

= 12 + 2 + 26 + 27 + 84 + 24 + 2 + 59

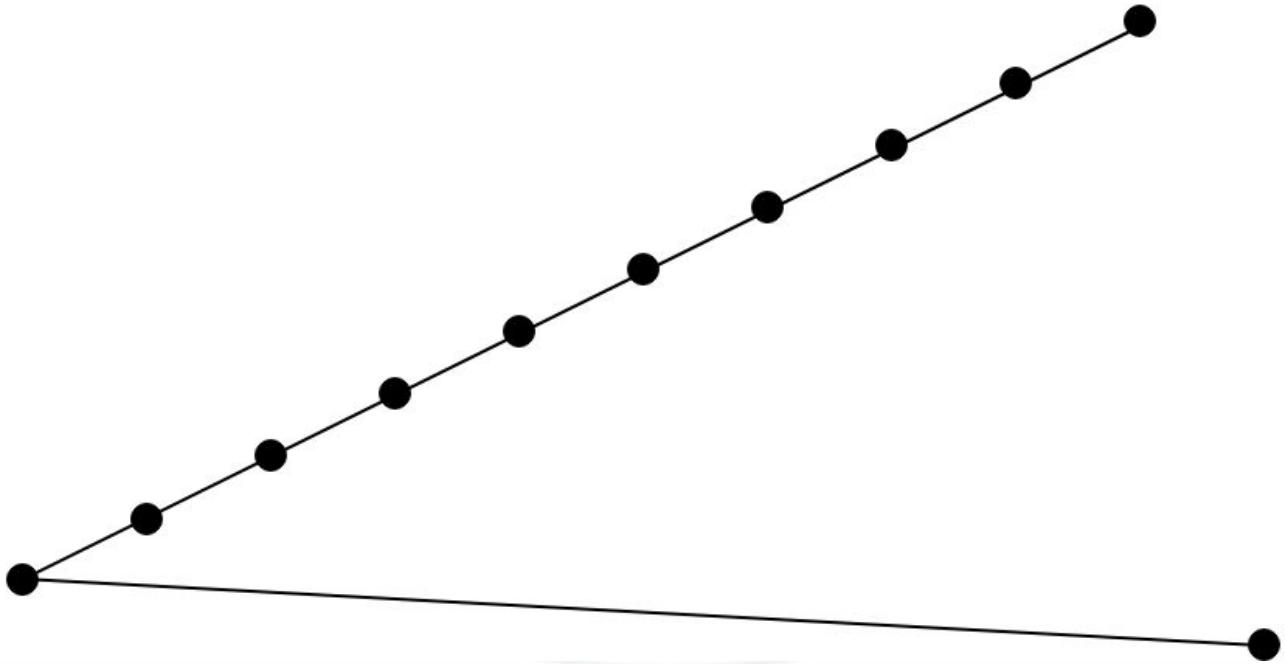
= 236

Problem

Consider a disk system with 100 cylinders. The requests to access the cylinders occur in following sequence- 4, 34, 10, 7, 19, 73, 2, 15, 6, 20

Assuming that the head is currently at cylinder 50, what is the time taken to satisfy all requests if it takes 1 ms to move from one cylinder to adjacent one and shortest seek time first policy is used?

4, 34, 10, 7, 19, 73, 2, 15, 6, 20



Total head movements incurred while servicing these requests

= (50 – 34) + (34 – 20) + (20 – 19) + (19 – 15) + (15 – 10) + (10 – 7) + (7 – 6) + (6 – 4) + (4 – 2) + (73 – 2)

= 16 + 14 + 1 + 4 + 5 + 3 + 1 + 2 + 2 + 71

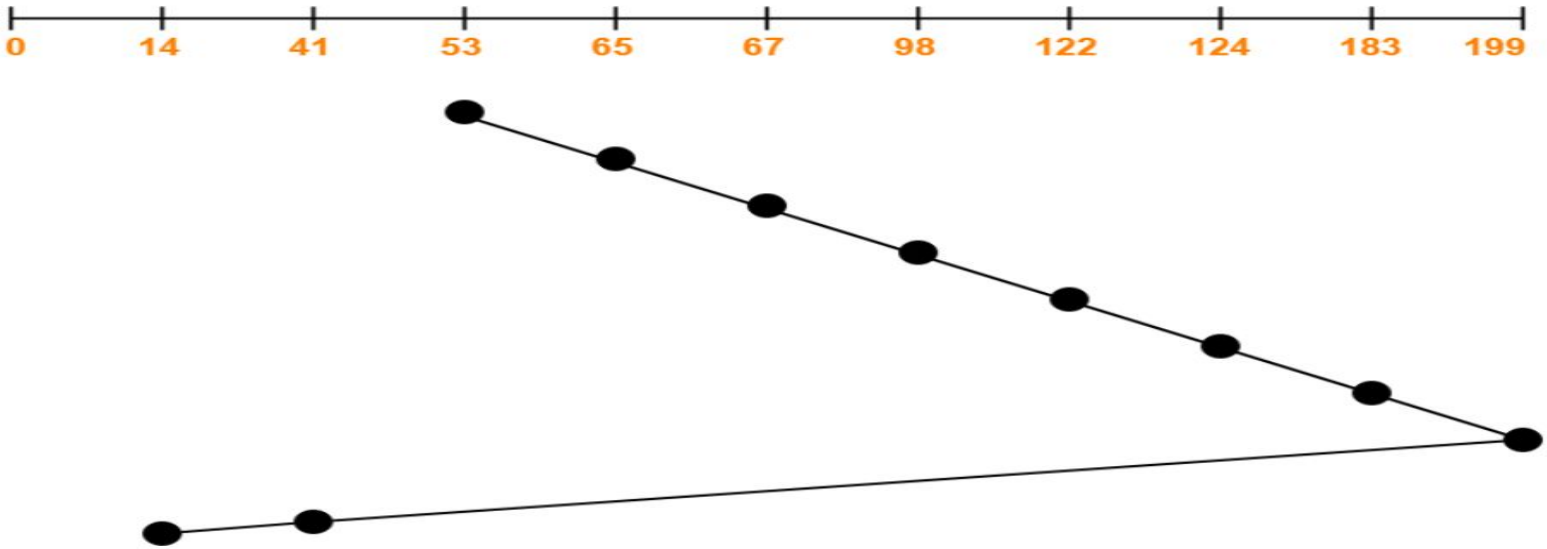
= 119

SCAN or Elevator Algorithm

- As the name suggests, this algorithm scans all the cylinders of the disk back and forth.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction and move towards the starting end servicing all the requests in between.
- SCAN called as **Elevator Algorithm** - its working resembles the working of an elevator.
- **Advantages-**
  - It is simple, easy to understand and implement.
  - It does not lead to starvation.
  - It provides low variance in response time and waiting time.
- **Disadvantages-**
  - It causes long waiting time for the cylinders just visited by the head.
  - It causes the head to move till the end of the disk even if there are no requests to be serviced.

Problem

Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The SCAN scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.



Total head movements incurred while servicing these requests

= (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (199 – 183) + (199 – 41) + (41 – 14)

= 12 + 2 + 31 + 24 + 2 + 59 + 16 + 158 + 27

= 331

Alternatively,

Total head movements incurred while servicing these requests

= (199 – 53) + (199 – 14)

= 146 + 185

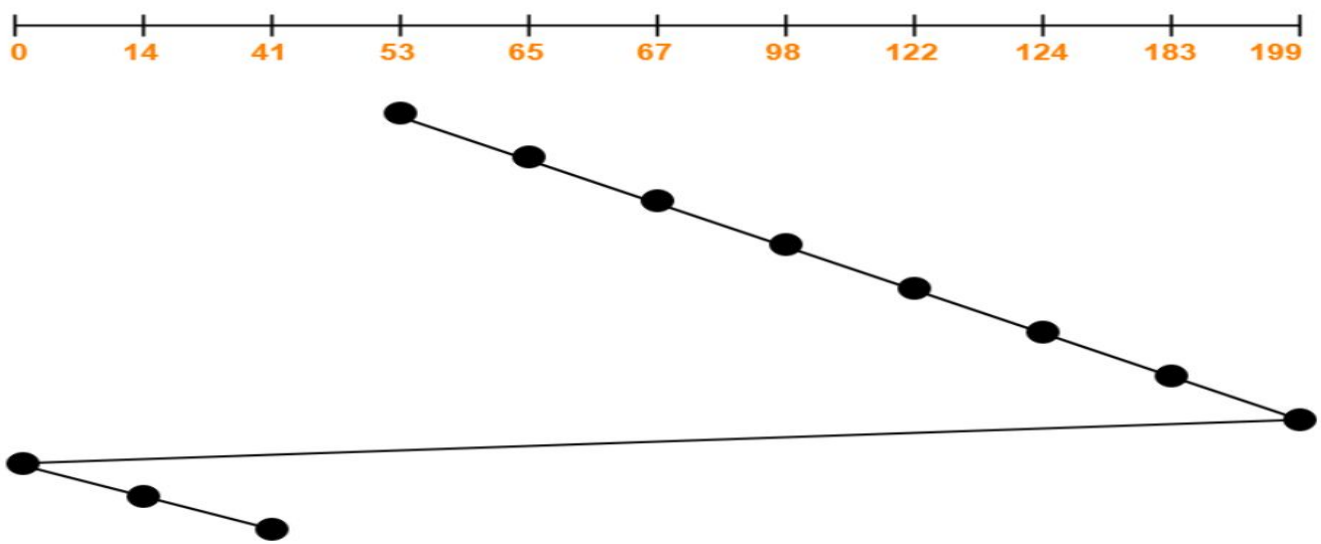
= 331

Circular-SCAN [C-SCAN]

- An improved version of the **SCAN Algorithm**.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction.
- It then returns to the starting end without servicing any request in between.
- The same process repeats.
- **Advantages**
  - Few requests are in front of the head, since these cylinders have recently been serviced. Hence provides a more uniform wait time.
- **Disadvantages**
  - It causes more seek movements as compared to SCAN Algorithm.
  - It causes the head to move till the end of the disk even if there are no requests to be serviced.

Problem

Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The C-SCAN scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.



Total head movements incurred while servicing these requests

= (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (199 – 183) + (199 – 0) + (14 – 0) + (41 – 14)

= 12 + 2 + 31 + 24 + 2 + 59 + 16 + 199 + 14 + 27

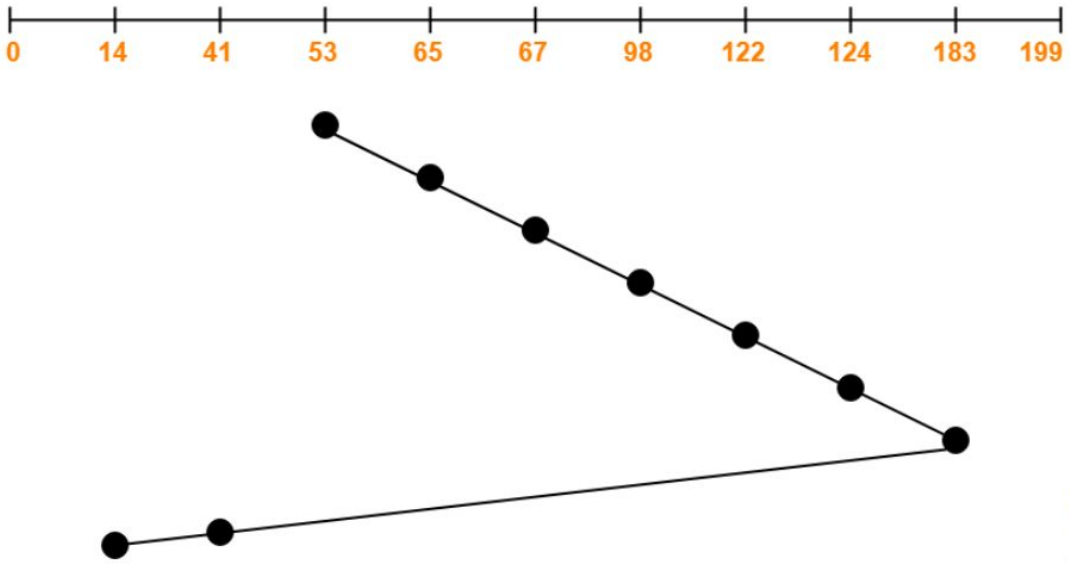
= 386

LOOK

- like SCAN but stops moving inwards (or outwards) when no more requests in that direction exist
- **SCAN vs LOOK**
  - SCAN Algorithm scans all the cylinders of the disk starting from one end to the other end even if there are no requests at the ends.
  - LOOK Algorithm scans all the cylinders of the disk starting from the first request at one end to the last request at the other end.
- **Advantages**
  - It provides better performance as compared to SCAN Algorithm.
  - It does not lead to starvation.
- **Disadvantages**
  - There is an overhead of finding the end requests.

Problem

Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The LOOK scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.



Total head movements incurred while servicing these requests

= (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (183 – 41) + (41 – 14)

= 12 + 2 + 31 + 24 + 2 + 59 + 142 + 27

= 299

Alternatively,

Total head movements incurred while servicing these requests

= (183 – 53) + (183 – 14)

= 130 + 169

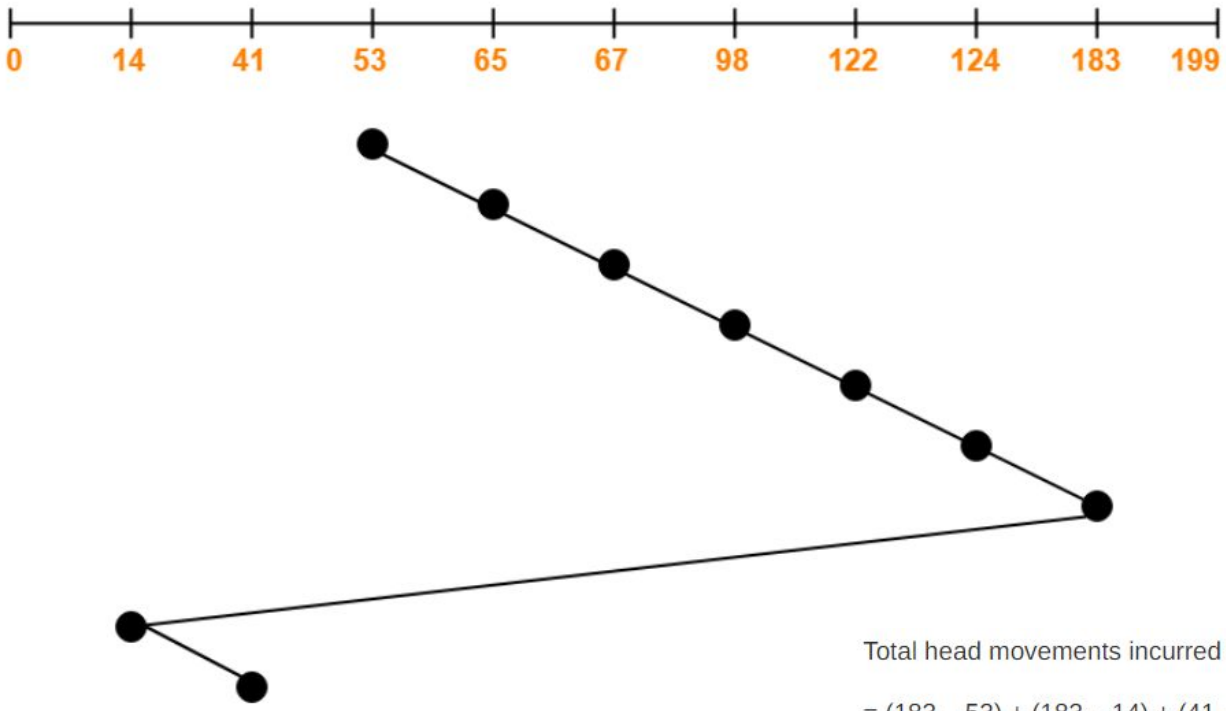
= 299

Circular LOOK [C-LOOK]

- Circular-LOOK Algorithm is an improved version of the LOOK Algorithm.
- Head starts from the first request at one end of the disk and moves towards the last request at the other end servicing all the requests in between.
- After reaching the last request at the other end, head reverses its direction.
- **Advantages**
  - It does not causes the head to move till the ends of the disk when there are no requests to be serviced.
  - It provides better performance as compared to LOOK Algorithm.
  - It does not lead to starvation.
- **Disadvantages**
  - There is an overhead of finding the end requests.

Problem

Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The C-LOOK scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.



Total head movements incurred while servicing these requests

= (183 – 53) + (183 – 14) + (41 – 14)

= 130 + 169 + 27

= 326

Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1,805. The queue of pending requests, in FIFO order, is: 2,069; 1,212; 2,296; 2,800; 544; 1,618; 356; 1,523; 4,965; 3,681 Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

a. FCFS
b. SSTF
c. SCAN
d. C-SCAN
e. LOOK
f. C-LOOK

Method	1	2	3	4	5	6	7	8	9	10	11	12	13	*Total*
FCFS	2150	2069	1212	2296	2800	544	1618	356	1523	4965	3681			13011
SSTF	2150	2069	2296	2800	3681	4965	1618	1523	1212	544	356			7586
SCAN	2150	2296	2800	3681	4965	**4999**	2069	1618	1523	1212	544	356		7492
C-SCAN	2150	2296	2800	3681	4965	**4999**	**0**	356	544	1212	1523	1618	2069	9917
LOOK	2150	2296	2800	3681	4965	2069	1618	1523	1212	544	356			7424
C-LOOK	2150	2296	2800	3681	4965	356	544	1212	1523	1618	2069			9137

Which one to choose?

- Performance depends on number and type of requests.
- SSTF over FCFS.
- SCAN, C-SCAN for systems that place a heavy load on the disk, as they are less likely to cause starvation.
- Default algorithms, SSTF or LOOK

UNIT V

- **Storage Management:** Mass Storage Structure: Overview – HDD Scheduling. File System: File Concept – Access Methods – Directory Structure – Protection. File System Implementation: File System Structure – File System Operations – Directory Implementation – Allocation Methods - Free Space Management. – Security : The Security Problem – program Threats - Case study: Linux System.

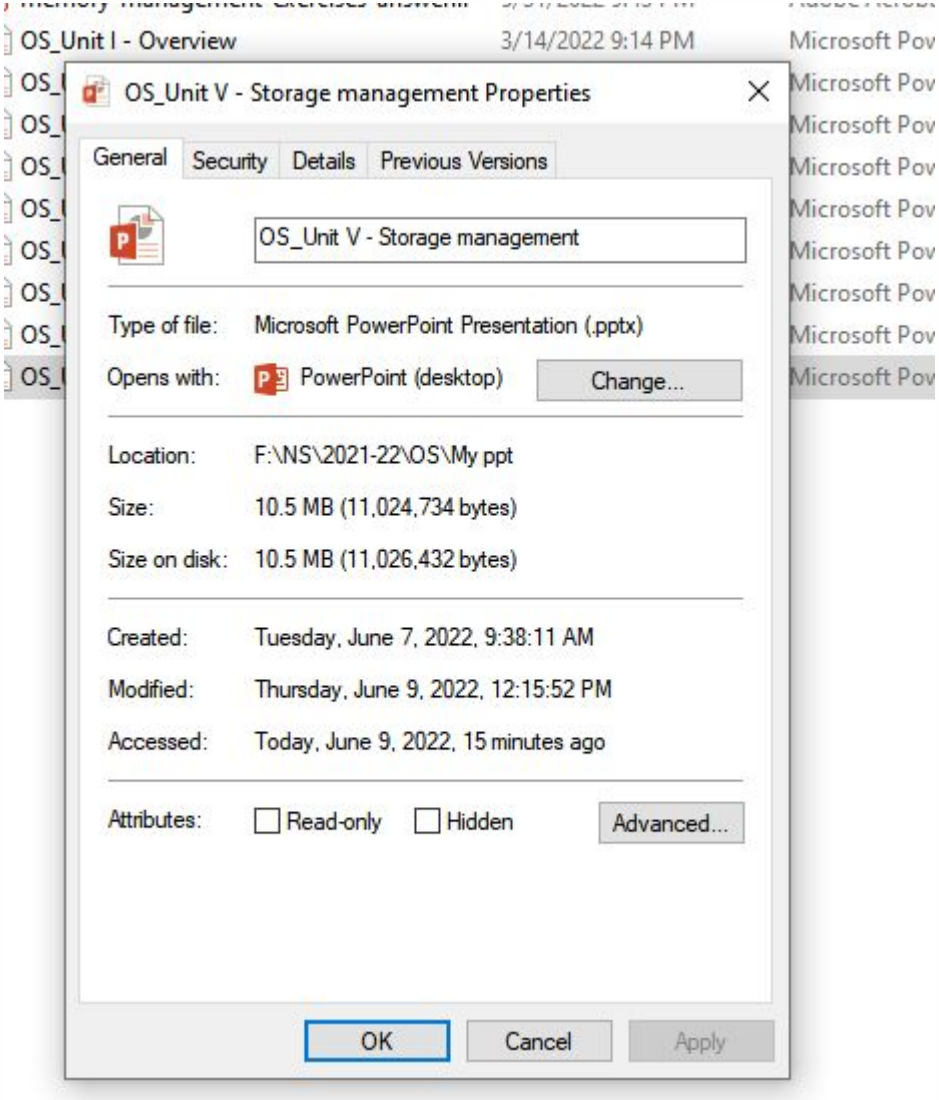
File Concept

- A file is a named collection of related information that is recorded on secondary storage
- Contiguous logical address space
- data cannot be written to secondary storage unless they are within a file
- Types:
  - Data
    - Numeric
    - Character
    - Binary
  - Program
- Contents defined by file's creator
  - Many types
    - **text file,**
    - **source file,**
    - **executable file**

File Attributes

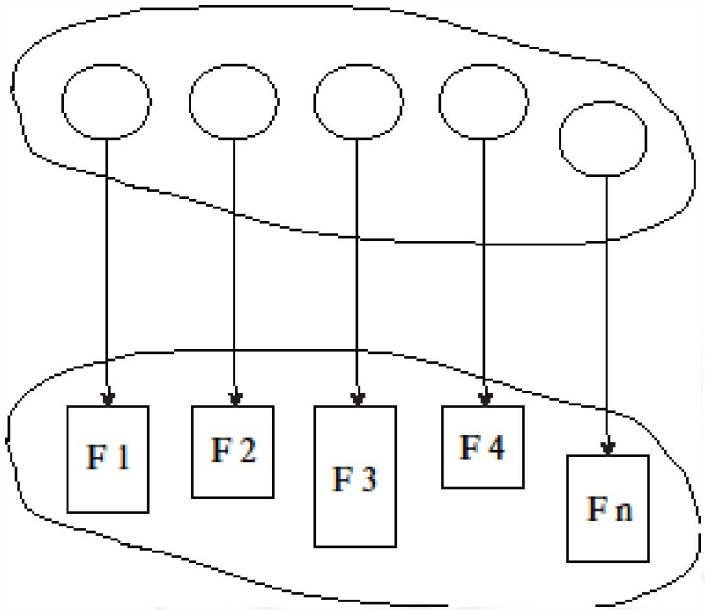
- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk
- Many variations, including extended file attributes such as file checksum

File info Window



Directory Structure

- A collection of nodes containing information about all files



- Both the directory structure and the files reside on disk

File Operations

- **Create**
- **Write –** at **write pointer** location
- **Read –** at **read pointer** location
- **Reposition within file - seek**
- **Delete**
- **Truncate**
- ***Open*** **(F_i)** – search the directory structure on disk for entry F_i, and move the content of entry to memory
- ***Close*** **(F_i)** – move the content of entry F_i in memory to directory structure on disk

Open Files

- Several pieces of data are needed to manage open files:
  - **Open-file table**: tracks open files
  - File pointer: pointer to last read/write location, per process that has the file open
  - **File-open count**: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it
  - Disk location of the file: cache of data access information
  - Access rights: per-process access mode information

File Locking

- Provided by some operating systems and file systems
  - Similar to reader-writer locks
  - **Shared lock** similar to reader lock – several processes can acquire concurrently
  - **Exclusive lock** similar to writer lock

- Mandatory or advisory:
  - **Mandatory** – access is denied depending on locks held and requested
  - **Advisory** – processes can find status of locks and decide what to do

File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

File Structure

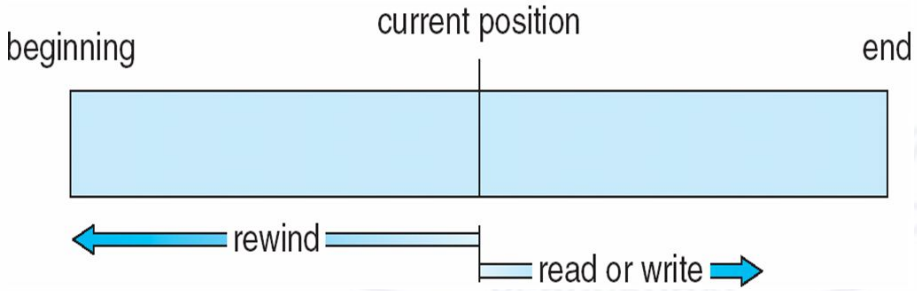
- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file
- Who decides:
  - Operating system
  - Program

Access Methods

- A file is fixed length **logical records**
- **Sequential Access**
- **Direct Access**
- **Other Access Methods**

Sequential Access

- Operations
  - **read next**
  - **write next**
  - **Reset**
- Figure



Direct Access

- Operations
  - **read *n***
  - **write *n***
  - **position to *n***
    - **read next**
    - **write next**
    - **rewrite *n***

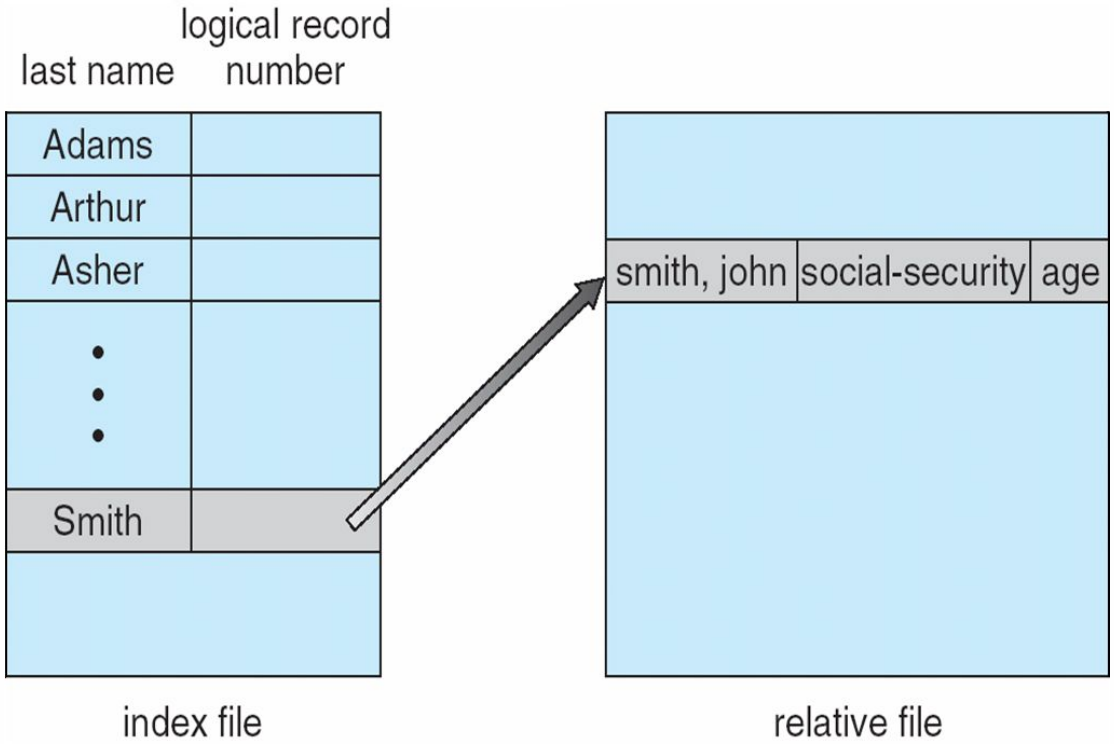
  n = **relative block number**

- Relative block numbers allow OS to decide where file should be placed

Other Access Methods

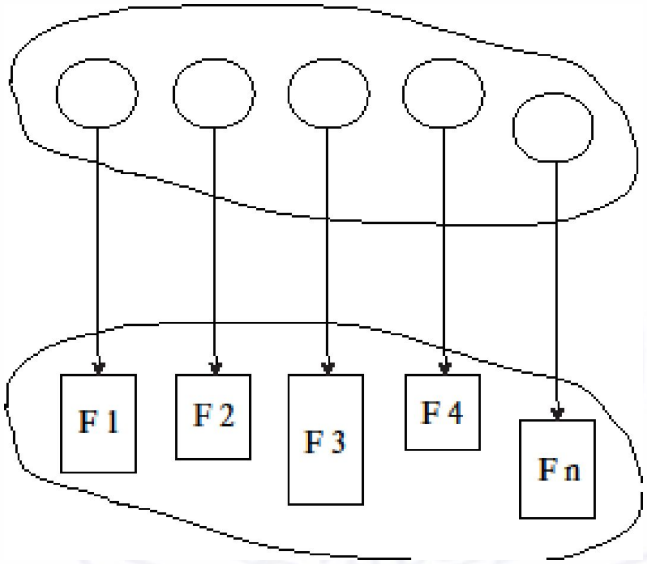
- Can be other access methods built on top of base methods
- General involve creation of an **index** for the file
- Keep index in memory for fast determination of location of data to be operated on (consider Universal Produce Code (UPC code) plus record of data about that item)
- If the index is too large, create an in-memory index, which an index of a disk index
- IBM indexed sequential-access method (ISAM)
  - Small master index, points to disk blocks of secondary index
  - File kept sorted on a defined key
  - All done by the OS
- VMS operating system provides index and relative files as another example (see next slide)

Example of Index and Relative Files



Directory Structure

- A collection of nodes containing information about all files



- Both the directory structure and the files reside on disk

Operations Performed on Directory

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

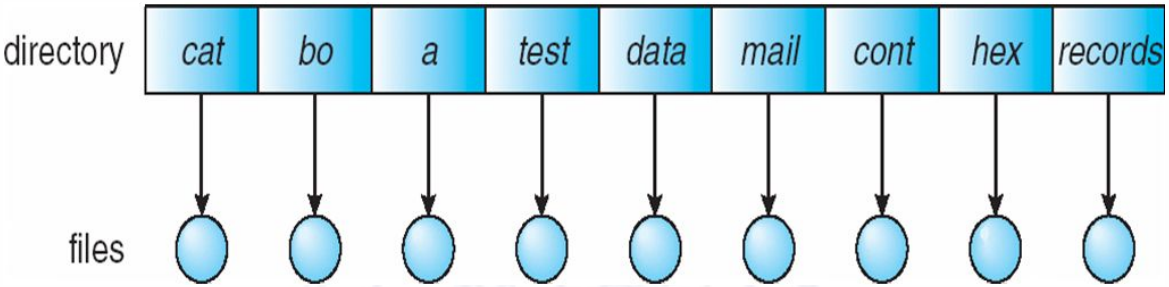
Directory Organization

The directory is organized logically to obtain

- Efficiency – locating a file quickly
- Naming – convenient to users
  - Two users can have same name for different files
  - The same file can have several different names
- Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)

Single-Level Directory

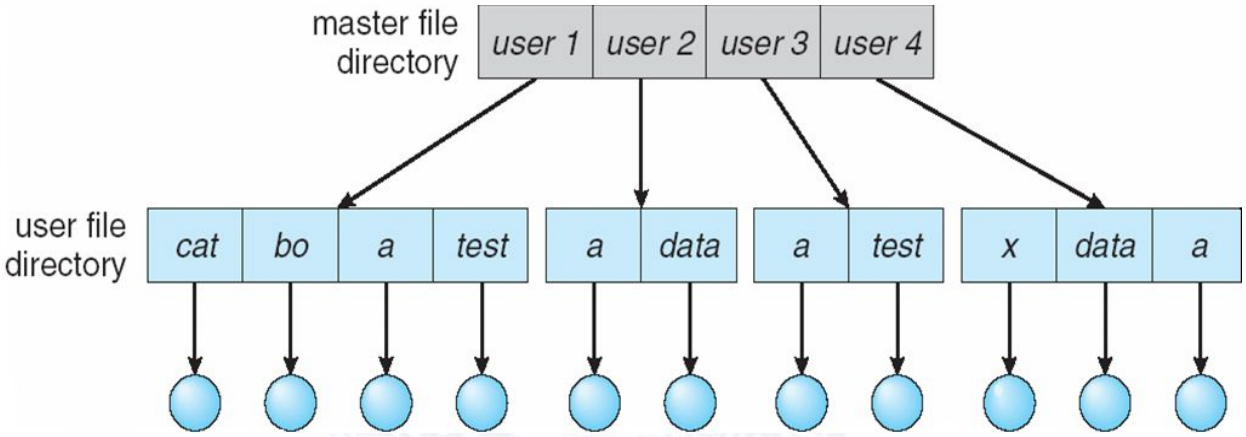
- A single directory for all users



- Naming problem
- Grouping problem

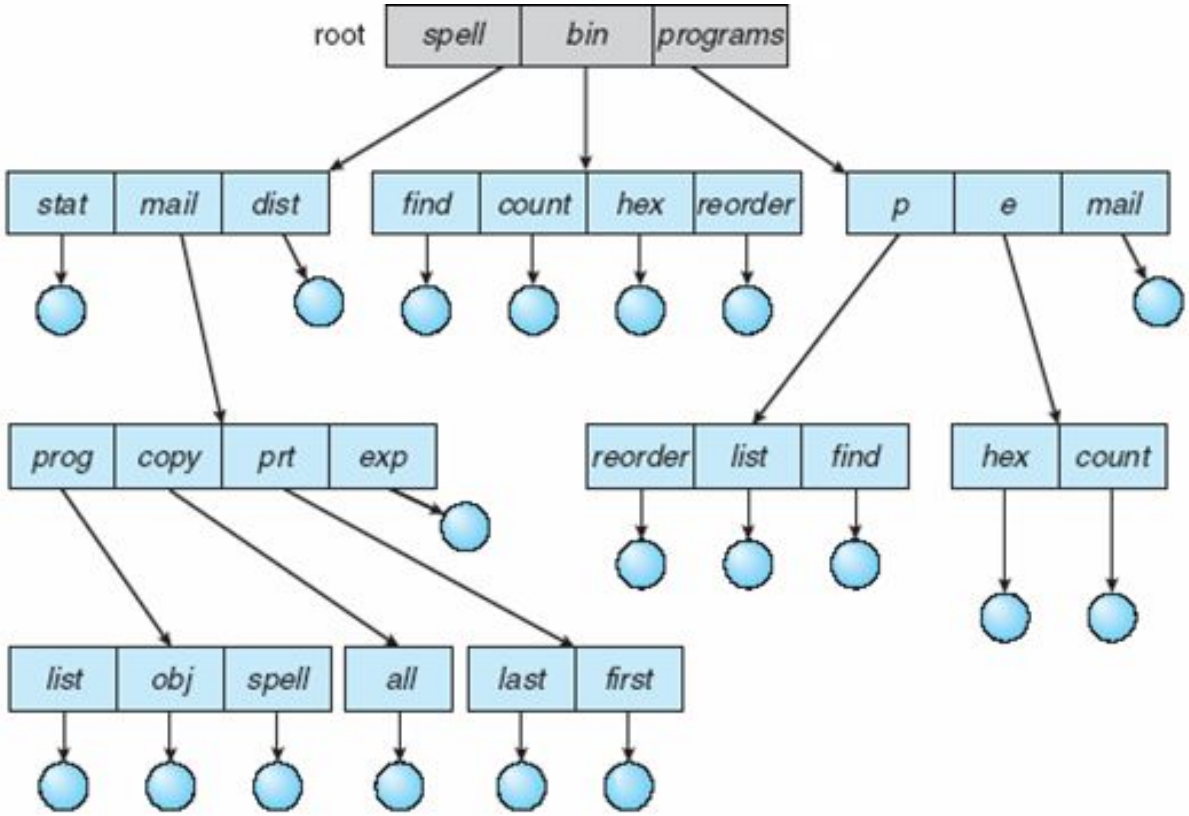
Two-Level Directory

- Separate directory for each user



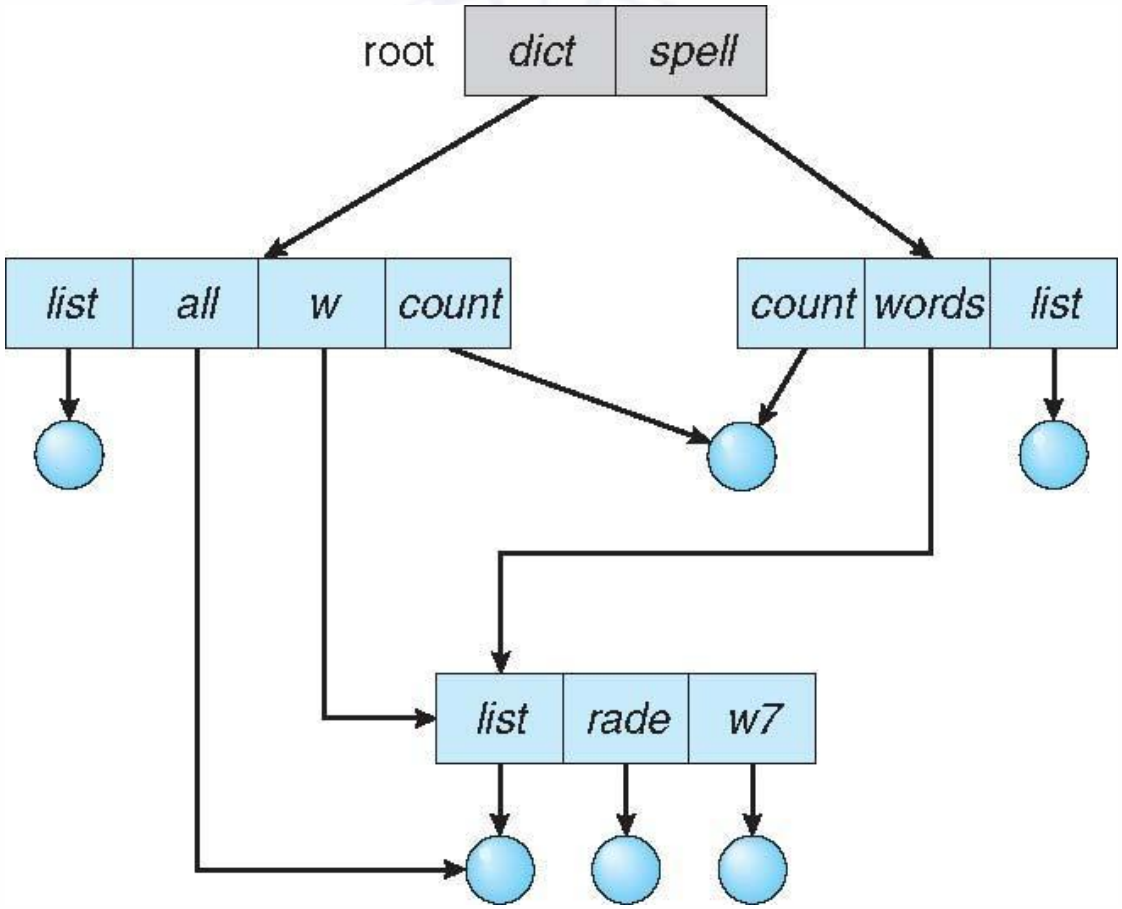
- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories



Acyclic-Graph Directories

- Have shared subdirectories and files
- An **acyclic graph**—that is, a graph with no cycles—allows directories to share subdirectories and files
- Example

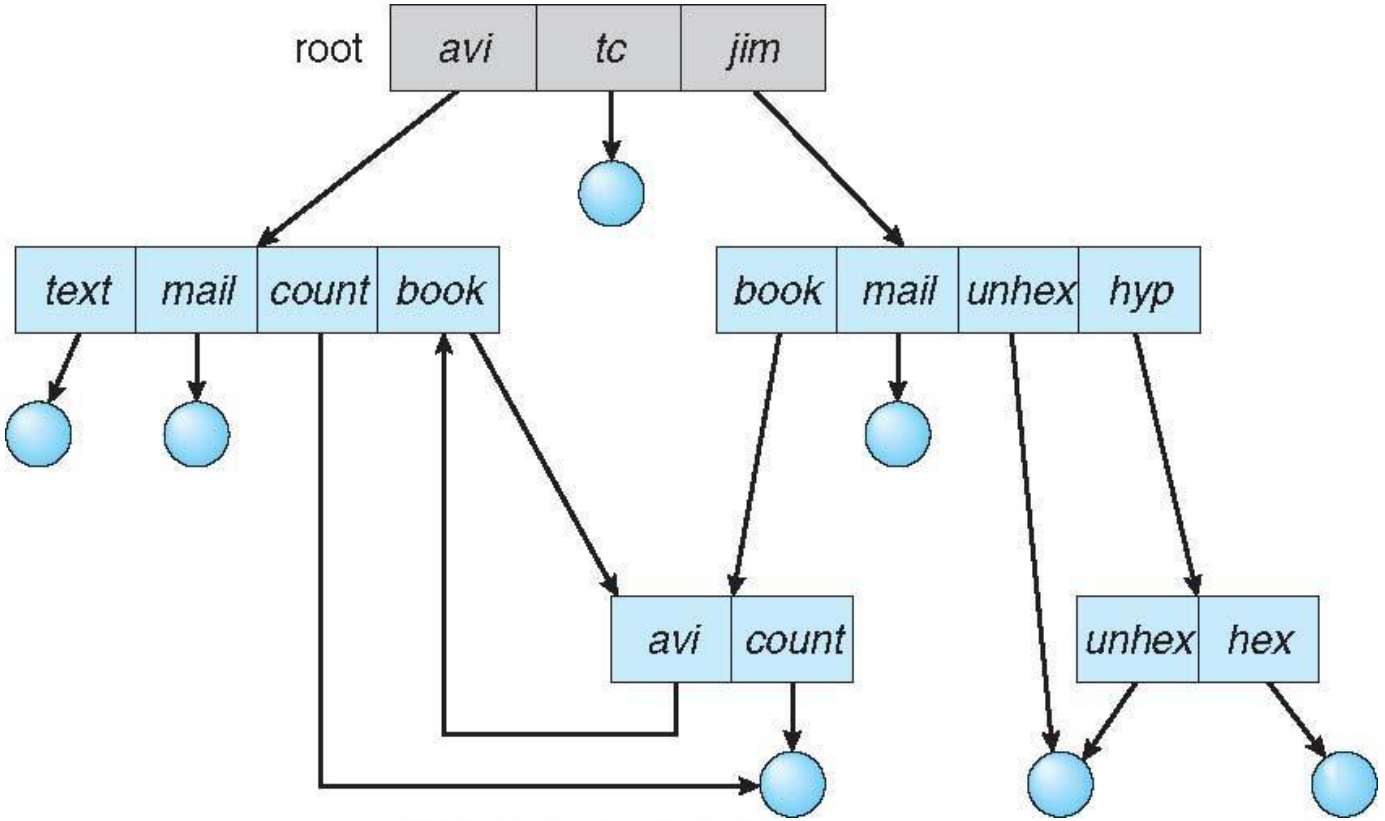


Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)
- If ***dict*** deletes **w/*list*** ⇒ dangling pointer

  Solutions:
  - Backpointers, so we can delete all pointers.
    - Variable size records a problem
  - Backpointers using a daisy chain organization
  - Entry-hold-count solution
- New directory entry type
  - **Link** – another name (pointer) to an existing file
  - **Resolve the link** – follow pointer to locate the file

General Graph Directory



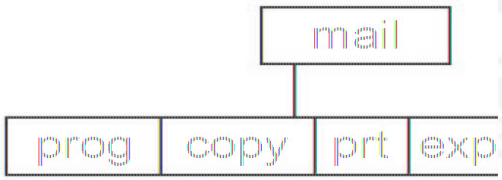
General Graph Directory (Cont.)

- How do we guarantee no cycles?
  - Allow only links to files not subdirectories
  - **Garbage collection**
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

Current Directory

- Can designate one of the directories as the current (working) directory
  - **`cd /spell/mail/prog`**
  - **`type list`**
- Creating and deleting a file is done in current directory
- Example of creating a new file
  - If in current directory is **`/mail`**
  - The command

    mkdir <dir-name>
  - Results in:



  - Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

Protection

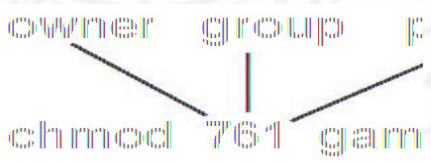
- File owner/creator should be able to control:
  - What can be done
  - By whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups in Unix

- Mode of access: read, write, execute
- Three classes of users on Unix / Linux

	RWX		
a) **owner access**	7	⇒	1 1 1
	RWX		
b) **group access**	6	⇒	1 1 0
	RWX		
c) **public access**	1	⇒	0 0 1

- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a file (say *game*) or subdirectory, define an appropriate access.



- Attach a group to a file

```
chgrp      G      game
```

A Sample UNIX Directory Listing

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Windows 7 Access-Control List Management