File System

File System

- It is the most visible aspect of operating system
- Provides mechanism for storage & access to both data and programs of OS
- Consists of two parts
  - collection of files
  - directory structure

File Concept

- A file is named collection of related information that is recorded on secondary storage
- It represents programs & data (numeric, alphabetic, alphanumeric & binary).
- It is a sequence of bits, bytes, lines or records, the meaning of which is defined by the file's creator & user
- A file has a certain defined structure which depends on its type (text, source, object, executable)

File Attributes

- Name – symbolic name in human-readable form
- Identifier – unique tag identifying the file within the 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

File Operations

- Create
  - Space in file system must be found for the file
  - An entry for new file must be made in directory
- Write
  - System call specifies name of file & information to be written on file
  - System searches for file location
  - System must keep write pointer to the location
- Read
  - System call specifies file name & location of next block of information
  - Read pointer kept at next location from where data is to be read

- Reposition within file
  - file seek; don't involve any I/O
- Delete
  - Search for the file to be deleted
  - Release file space and erase directory entry
- Truncate
  - Erase contents but keep the file structure
  - All attributes remains same but length becomes zero
- Other operations on file
  - Append – writing at the end
  - Copy – copying file contents to another file
  - Get or set attributes of files

File Table

- Operating system keeps a small table containing information about all open files
- When a file operation is requested, the file is specified via an index into this table, so no searching is required
- When a file is closed by the process the OS remove the entry from the File Table
- The open file table also has an open count associated with each file, indicating the number of processes that have the file open

- Two levels of internal tables are used to maintain information about open files
  - Per process table
    - tracks all files that a process has open
    - maintains process specific information such as access rights to file, pointer position in file etc.
    - Every entry points to corresponding system wide table entry
  - System wide table
    - contains process independent information such as file location, access date, file size etc

- Information associated with open files
  - File pointer
    - unique to each process which gives location of cursor in file
  - File open count
    - count no. of processes that have opened a file
  - Disk location of the file
    - information to locate file on disk kept in memory
  - Access rights
    - a file is opened in a particular access mode by a process

Open File Locking

- Provided by some operating systems and file systems
- Two types of locks
  - Shared lock – several processes can acquire lock concurrently
  - Exclusive lock – only one process can acquire such lock at a time
- Mandatory or advisory:
  - Mandatory – OS ensures locking integrity; access is denied depending on locks held and requested
  - Advisory – software developers need to ensure that locks are appropriately acquired and released

File Types

- If an OS recognizes the type of file, it then operates in appropriate ways.
- Name is split into 2 parts :
  - Name
  - Extension
- OS uses the extension to indicate the type of the file and the type of the operations that can be done on the file

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, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

File Structure

- Different file types have different file structures
- Ex. executable file structure stores where to load file in memory, what the location of first instruction is
- Multiple file structures results in greater size of OS as it needs to have code to handle different file structures
- Different file implementations-
  - Minimum number of file structures must be supported by OS
  - OS treats all files as same. Each application code must include its own code to interpret an input file as to the appropriate structure.

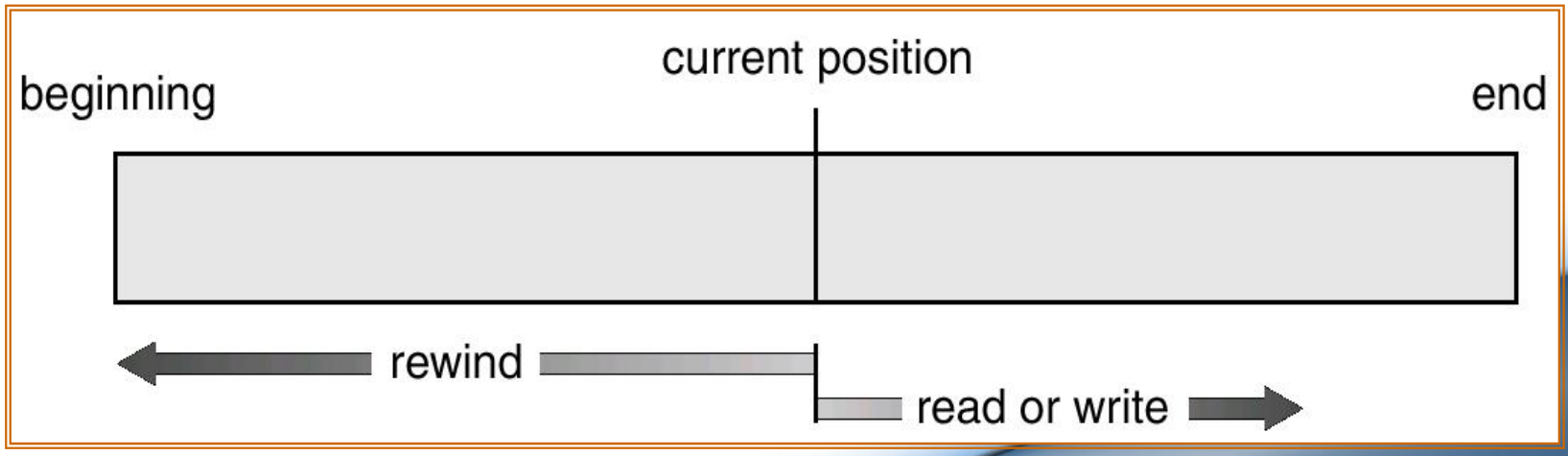
Access Methods

- File access mechanism refers to the manner in which the records of a file may be accessed.
- Types :
  - Sequential access
  - Direct/Random access
  - Indexed sequential access

Sequential Access

- Information in the file is processed in order, one record after the other
- E.g. – Editors & compilers access files in this fashion.
- read next – reads the next portion of the file and automatically advances a file pointer
- write next – appends to the end of the file and advances the end of the newly written data (new end of file)
- For example magnetic tapes store information in sequential manner and retrieval is performed in the same manner

Sequential Access

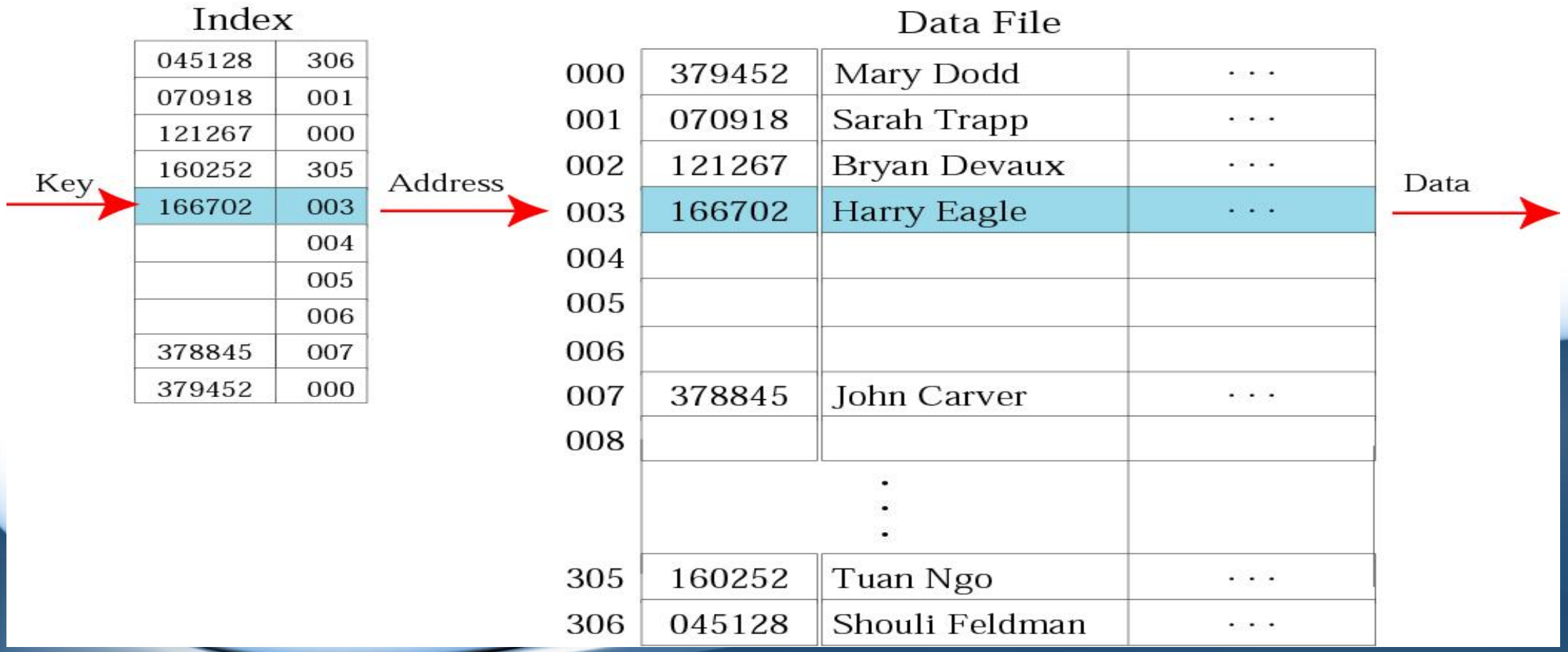


Direct Access

- A file is made up of fixed length logical records that allow programs to read and write records rapidly in no particular order
- It is based on the disk model of the file and it allows random access to any block
- For direct access the file is treated as a numbered sequence of blocks or records
- Hence we may read a block 12, then block 57 and then write block 6
- There is no restriction on the order of reading or writing for a direct access file
- read n and write n instead of read next and write next

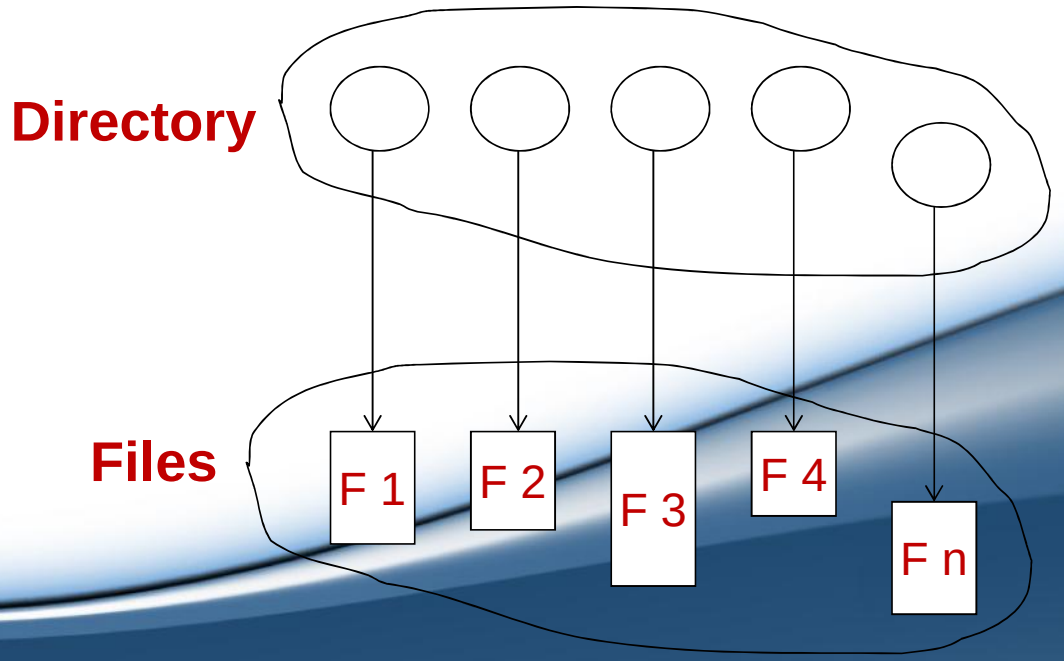
Indexed Sequential Access

- Built up on base of sequential access
- An index is created for each file which contains pointers to various blocks
- Index is searched sequentially and its pointer is used to access the file directly


Index
045128 306
070918 001
121267 000
160252 305
166702 003
004
005
006
378845 007
379452 000
Key
Address
Data File
000 379452 Mary Dodd · · ·
001 070918 Sarah Trapp · · ·
002 121267 Bryan Devaux · · ·
003 166702 Harry Eagle · · ·
004
005
006
007 378845 John Carver · · ·
008
305 160252 Tuan Ngo · · ·
306 045128 Shouli Feldman · · ·
Data

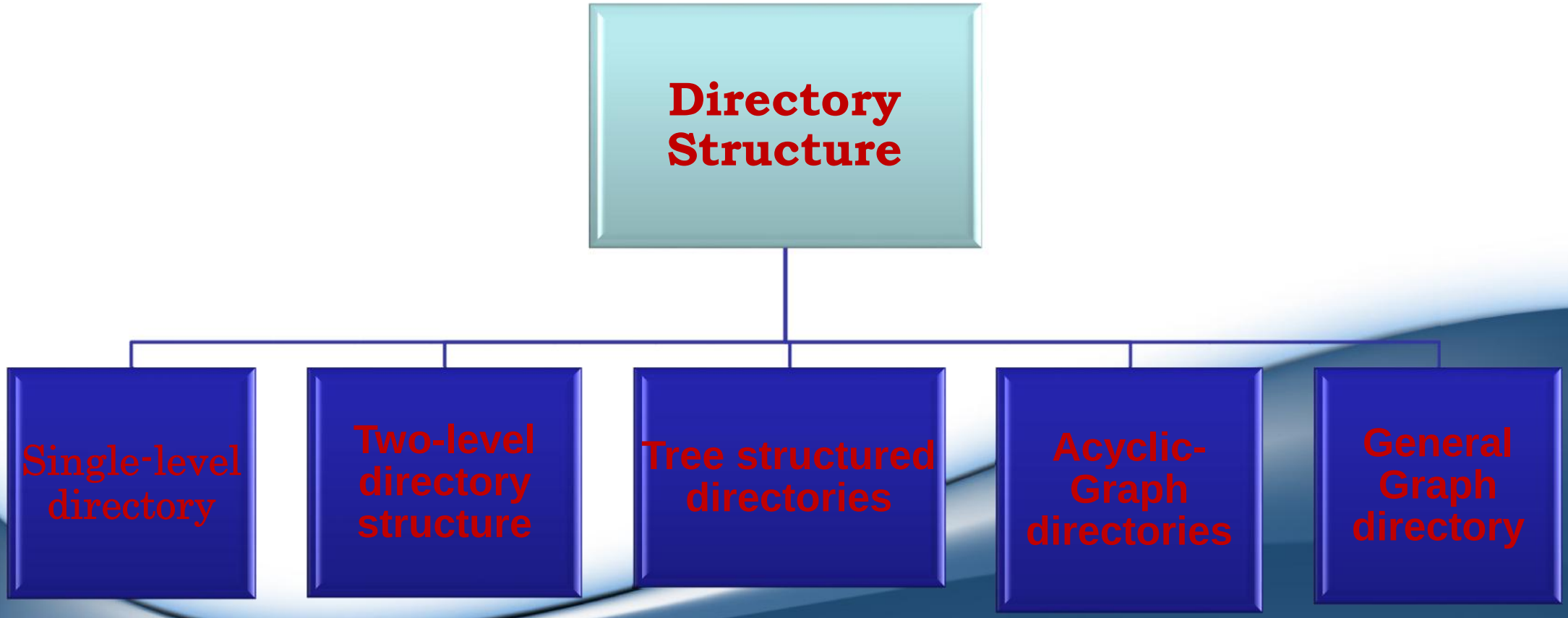
Directory Systems

- Similar to symbol table that translates file names into their directory entries
- Can be organized in many ways



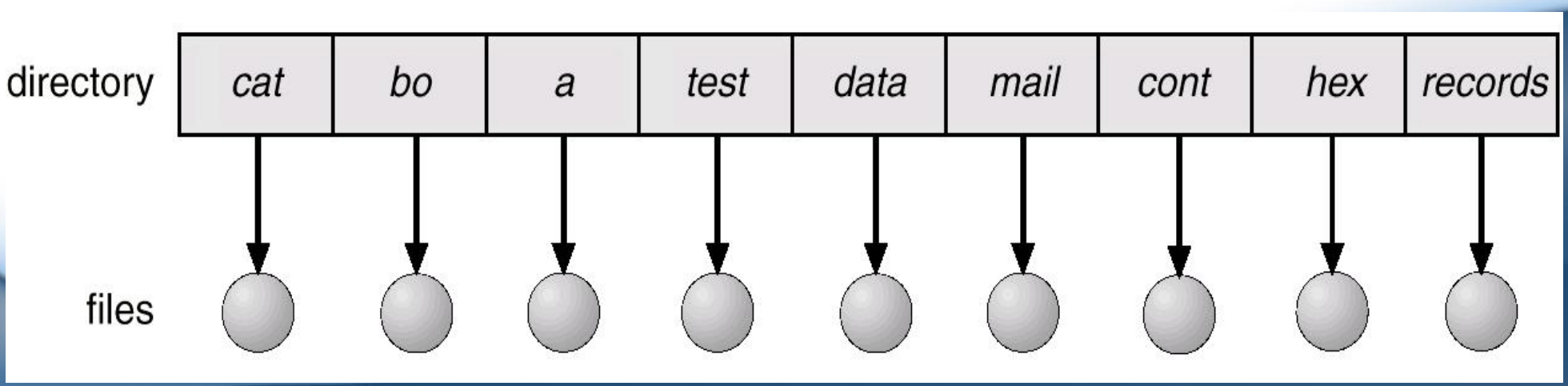
Directory Systems

- Operations performed on directories :
  - Search
  - Create
  - Delete
  - List a directory : list the files in a directory & the contents of the directory entry for each file.
  - Rename
  - Traverse the file system


Directory Structure
Single-level directory
Two-level directory structure
Tree structured directories
Acyclic-Graph directories
General Graph directory

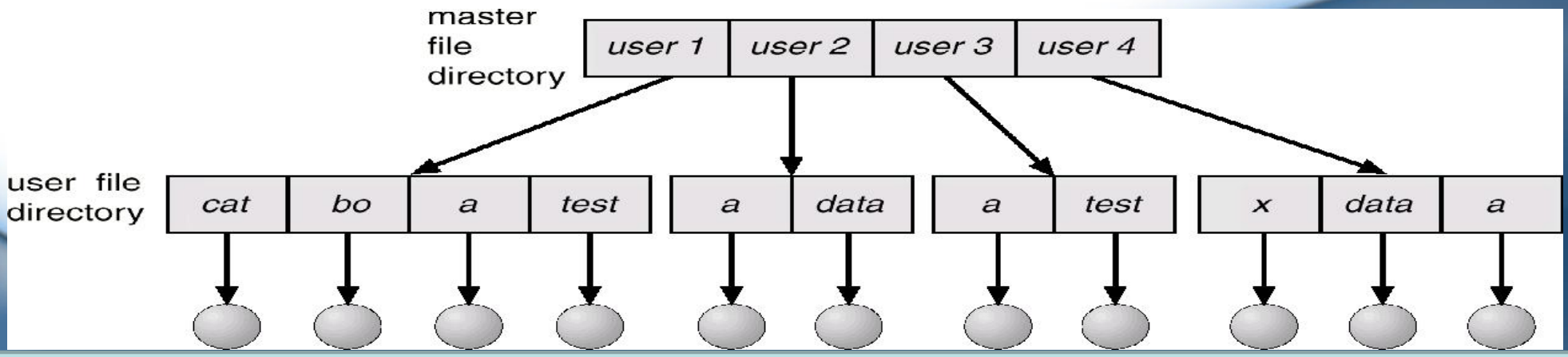
Single-Level Directory

- All files are contained in the same directory
- A single directory for all users
- Limitation – Naming problem

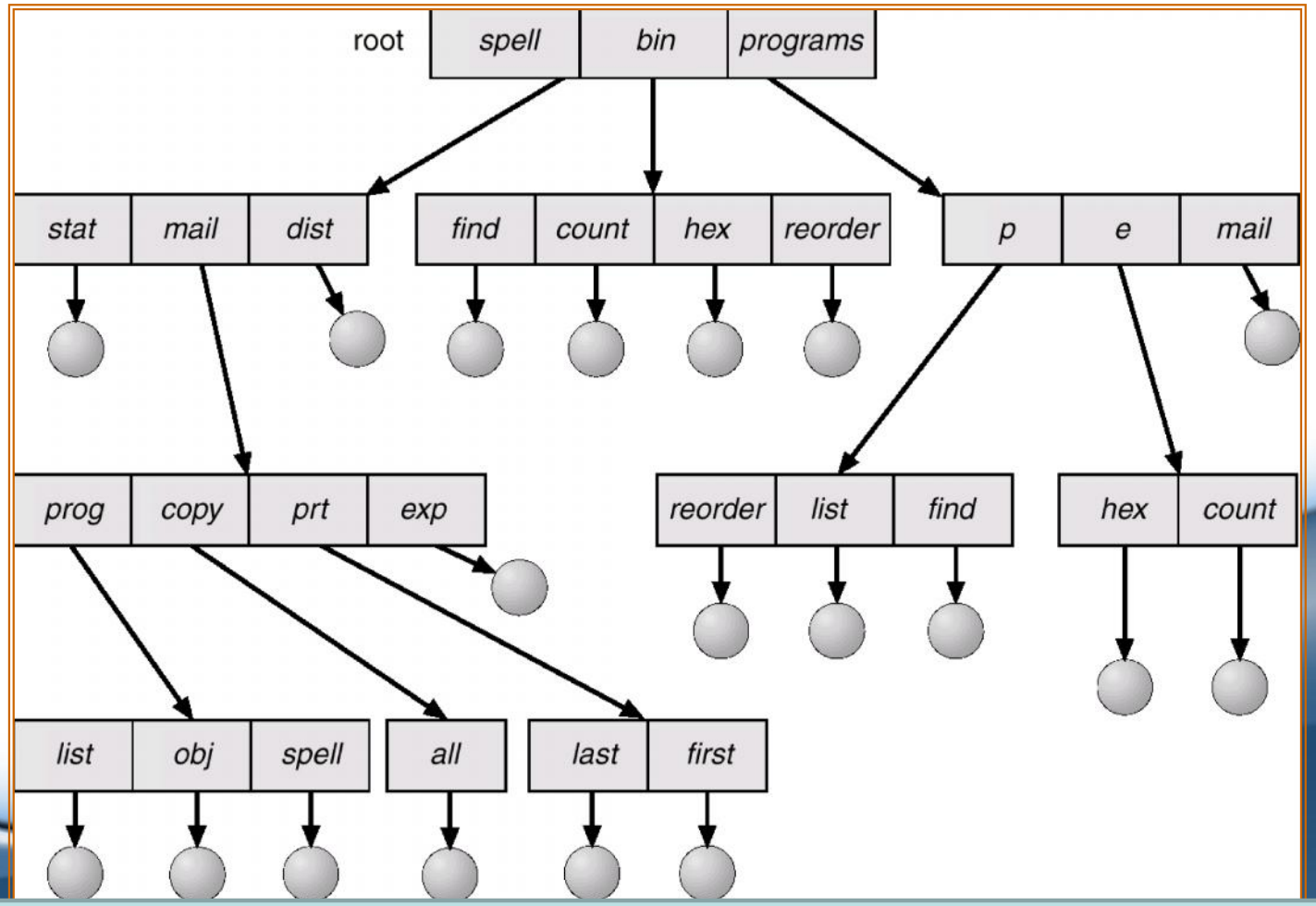


Two-Level Directory

- Each user has his own user file directory (UFD)
- When user logs in, systems Master file directory (MFD) is searched
- MFD
  - indexed by user name or account no.
  - Each entry points to the UFD for that user.



Tree Structured Directories



Tree Structured Directories

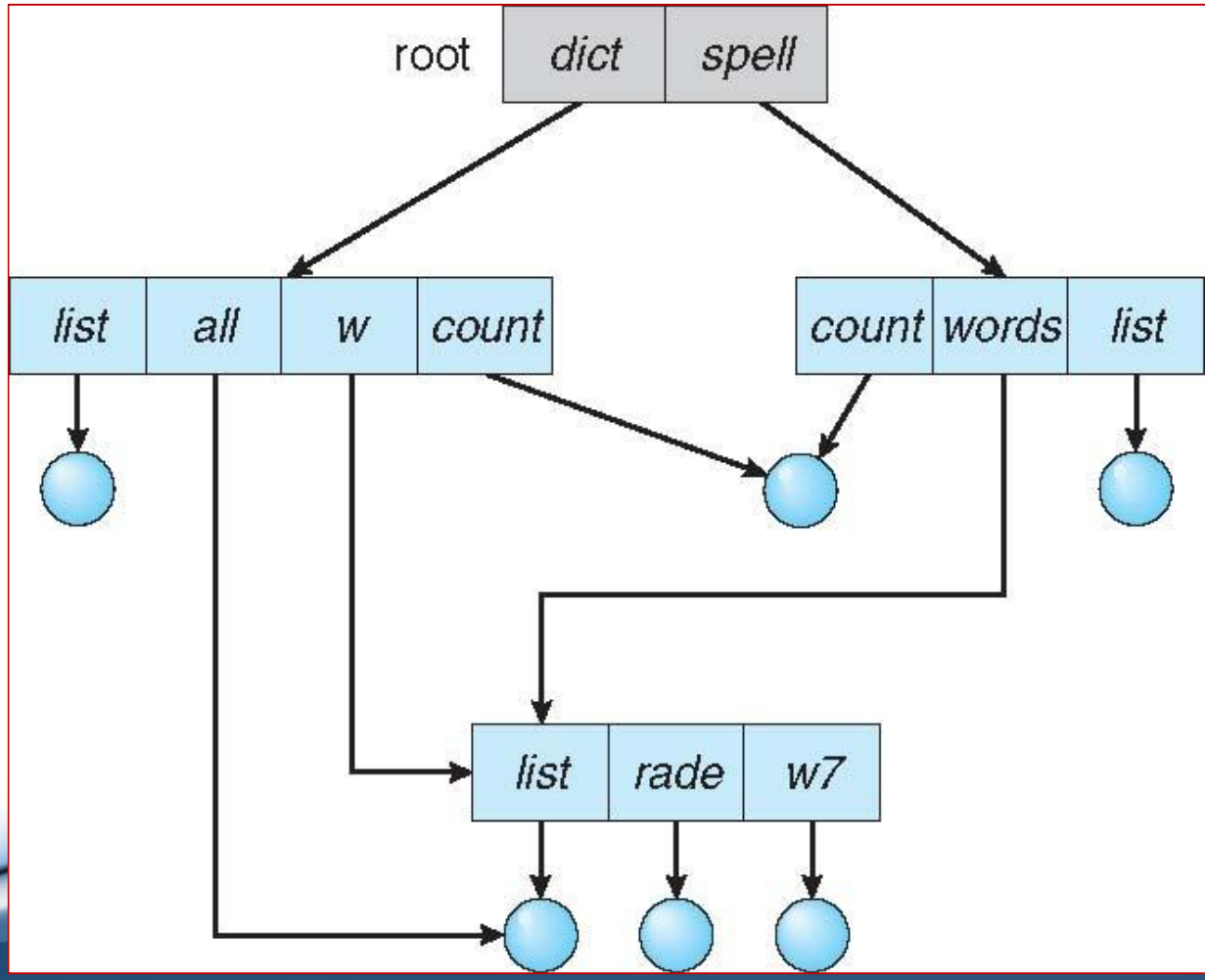
- Every file in a system has a unique path name
- One bit in each directory defines the entry as a file(0) or as a subdirectory(1)
- Special system calls are used to create, delete or change current directories.
- Path names can be of 2 types
  - Absolute path
    - Begins at the root & follows the path down to the specified file
  - Relative path
    - Defines path from current directory

- Deletion of directory
  - First delete all the files and subdirectories in specific directory to delete that directory
  - Use a command eg, rm in UNIX which will first empty the directory and then remove directory.

    rm <file-name>

Acyclic – Graph Directories

Allows shared subdirectories and files



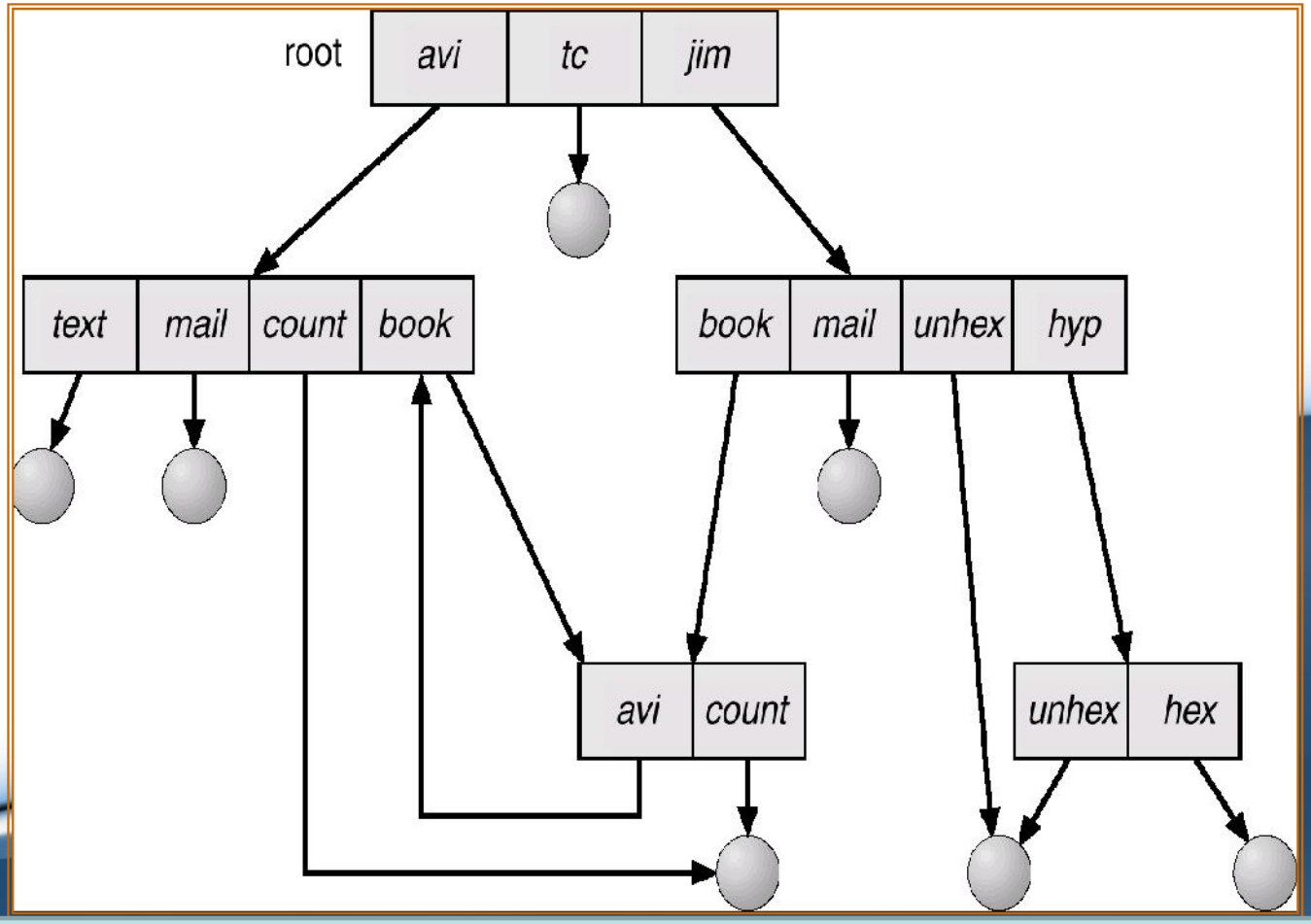
Acyclic – Graph Directories

- Two programmers working on a joint project, would like to have files associated with that project to be shared in a subdirectory
- As both the programmers would like to have subdirectory under their own directories and hence the common subdirectory should be shared
- The acyclic graph is a natural generalization of the tree-structured directory scheme
- An acyclic graph (a graph with no cycles) allows directories to share subdirectories and files

- There exists only one copy of the file and any changes made by one is immediately visible to the other
- Sharing is particularly for subdirectories; a new file created by one person will automatically appear in all the shared subdirectories
- Shared files and subdirectories can be implemented in several ways
- A common way adopted in UNIX system is to create a new directory entry called link
- Link is effectively a pointer to another file or subdirectory
- An acyclic-graph directory structure is more flexible than a simple tree structure, but is more complex

General Graph Directory

When links are added to an existing tree-structured directory, a general graph structure can be created



File Protection

- File owner/creator should be able to control - what can be done, by whom
- Protection mechanism provides controlled access by limiting the types of file access that can be made
- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

Access Control List (ACL)

- Access control list (ACL) specifies user names and type of access allowed for each user
- OS checks this ACL, when a user requests an access to a particular file or directory and allows/ disallows access depending on the ACL
- Disadvantage
  - Length of ACL (if want to provide read access to all users, must list all in ACL)
  - Tedious process especially if list of users not known in advance
  - Directory entry has to be of variable length; complicated space management

Access Control List (ACL)

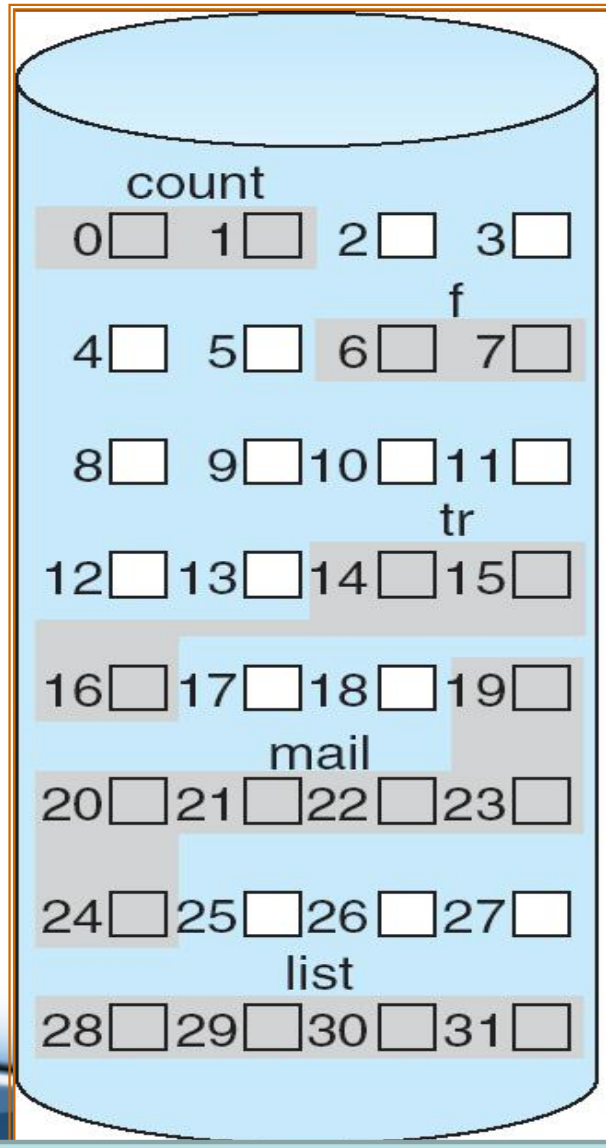
- To reduce the length of the ACL, many systems recognize three classes of users
  - Owner: One who created the file
  - Group: A set of users who are sharing the file
  - Universe: All the other users in the system
- Another approach to the protection problem is to associate a password with each file.
  - Effective if password are chosen randomly & changed often
  - Disadvantage:
    - Remembering all passwords is cumbersome.
    - Keeping same password for all files also unsafe.

Allocation Methods

- How to allocate space to files so that disk space is utilized effectively and files can be accessed quickly
- Three major methods of allocating disk space are in wide use
  1. Contiguous allocation
  2. Linked allocation
  3. Indexed allocation
- Each method has its own advantages and disadvantage

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk
- Disk addresses define a linear ordering on the disk
- With this ordering, assuming that only one job is accessing the disk, there is no head movement required, even if required it is to the next track
- Hence the disk seek time is minimal
- Directory entry for each file has address of starting block and length of area allocated for this file
- File access : both sequential as well as direct access is supported



directory

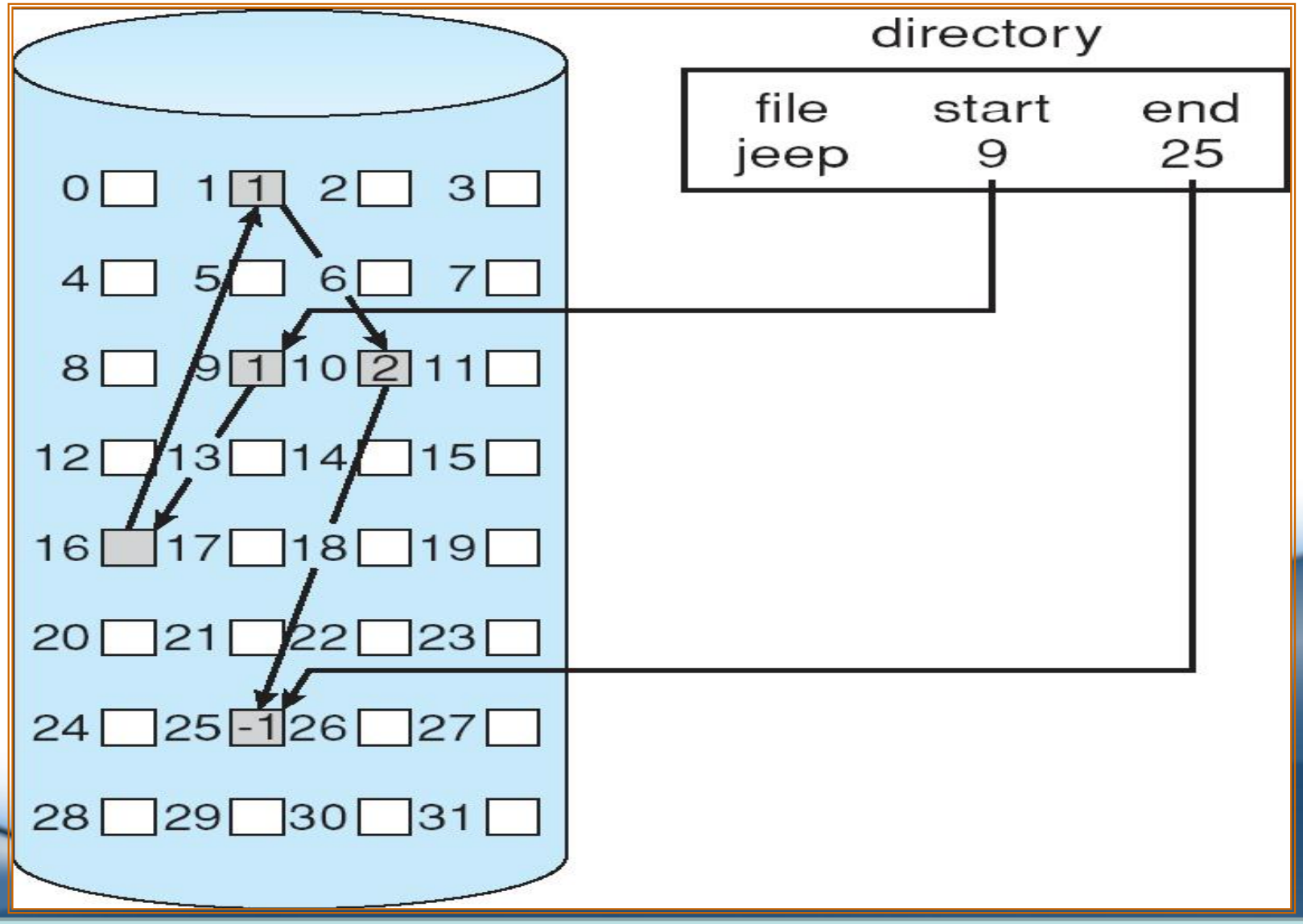
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Problems in Contiguous Allocation

- Finding free space for a new file
  - Dynamic storage-allocation problem : how to satisfy a request for size n from a list of free holes – best fit OR first fit
  - They are efficient but not best in terms of storage utilization
  - All these algorithms suffer from the problem of external fragmentation
- Determine how much space is needed for a file
  - When file created, total amount of space it will need must be found & allocated
  - Sometimes difficult to estimate (Size of output file ?)
  - What is user wants to extend file

Linked Allocation

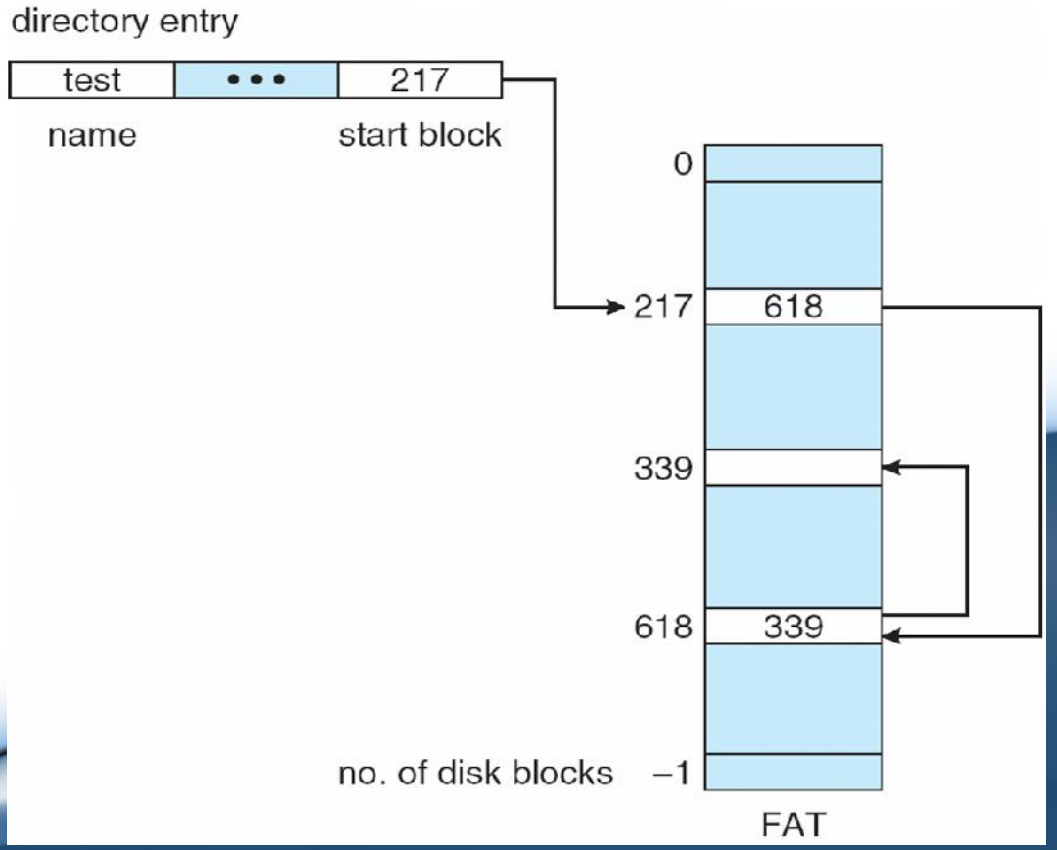
- Each file is a linked list of blocks ; blocks may be scattered anywhere on the disk
  - Directory contains a pointer to first & last blocks of file
  - Each block contains pointer to next block
  - File ends at nil pointer
  - No external fragmentation
  - Free space management system called when new block needed; file can continue to grow
  - Disadvantage :
    - Used only for sequential access files.
    - Space required for pointer


directory
file
start
end
jeep
9
25
0
1
1
2
3
4
5
6
7
8
9
1
10
2
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
-1
26
27
28
29
30
31

- No direct access - To find i^{th} block , must start from the beginning and follow all pointers until it reaches i^{th} block
- Each file requires more space than the actual required one, to store all pointers.
- Solution- collect blocks into multiples, called clusters and to allocate clusters rather than blocks. But increases internal fragmentation
- Reliability lost if pointer lost or damaged
- Locating a block can take many I/Os and disk seeks – solution file allocation table

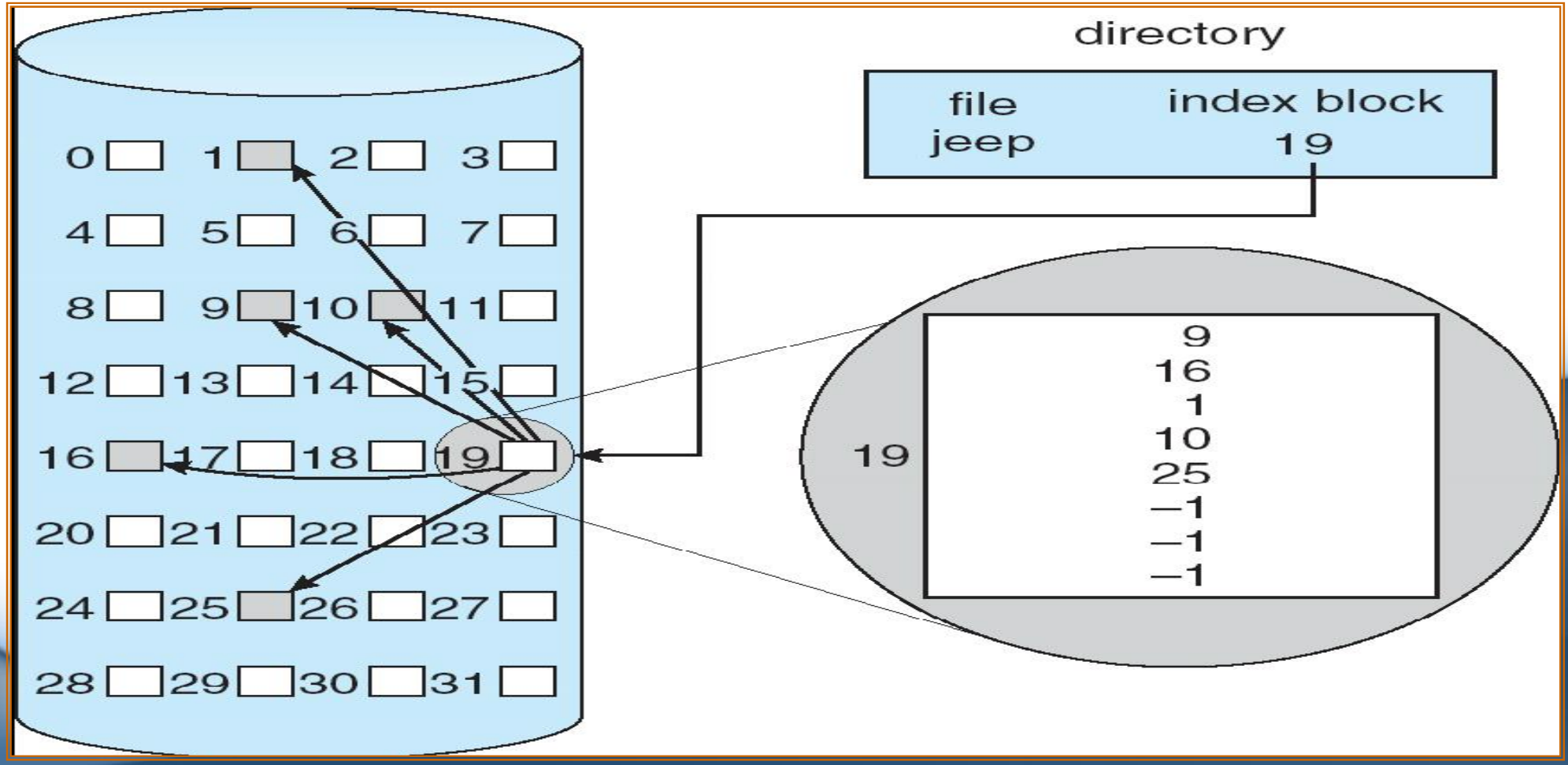
File-Allocation Table

- FAT (File Allocation Table)
  - It has 1 entry for each disk block & is indexed by block number
  - Much like a linked list, but faster on disk and cacheable
  - New block allocation simple



Indexed Allocation

- Bring all pointers together into one location : the index block
- Each file has its own index block, which is an array of disk-block addresses
- Directory block contains address of the index block
- When a file is created, all pointers in the index block are set to nil
- When i^{th} block is written, its address is put in the i^{th} index block entry
- Supports direct access
- Does not suffer form external fragmentation


directory
file
jeep
index block
19
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31
19
9
16
1
10
25
−1
−1
−1

Comparison of File Allocation Methods

- Contiguous allocation requires just one disk access for both sequential and direct access
- Linked allocation is good for sequential access. Not suitable for direct access - for accessing i^{th} block, it might require i disk accesses
- Indexed allocation more complex – if index block already in memory, then access can be made directly. Else requires two accesses – to read index block and read desired data block

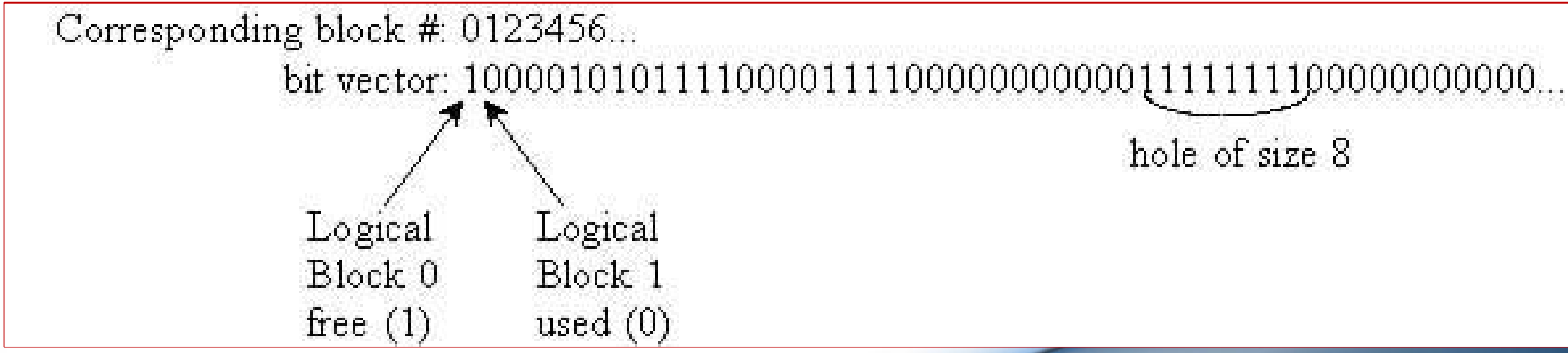
Free Space Management

- To keep track of free disk space, the system maintains a free-space list.
- The free – space list records all free disks blocks, those not allocated to some file or directory.
- Ways to implement free space list
  - Bit Vector
  - Linked List
  - Grouping
  - Counting

Bit Vector / Bit Map

- Each block is represented by 1 bit.
- Free block → bit=1; Allocated block→ bit=0
- Advantage:
  - It is relatively simple and efficient in finding the first free blocks or n consecutive free blocks on the disk.
  - Easy to get contiguous files
- Disadvantage:
  - May require special hardware support.
  - Bit map requires extra space, if the disk size is large.

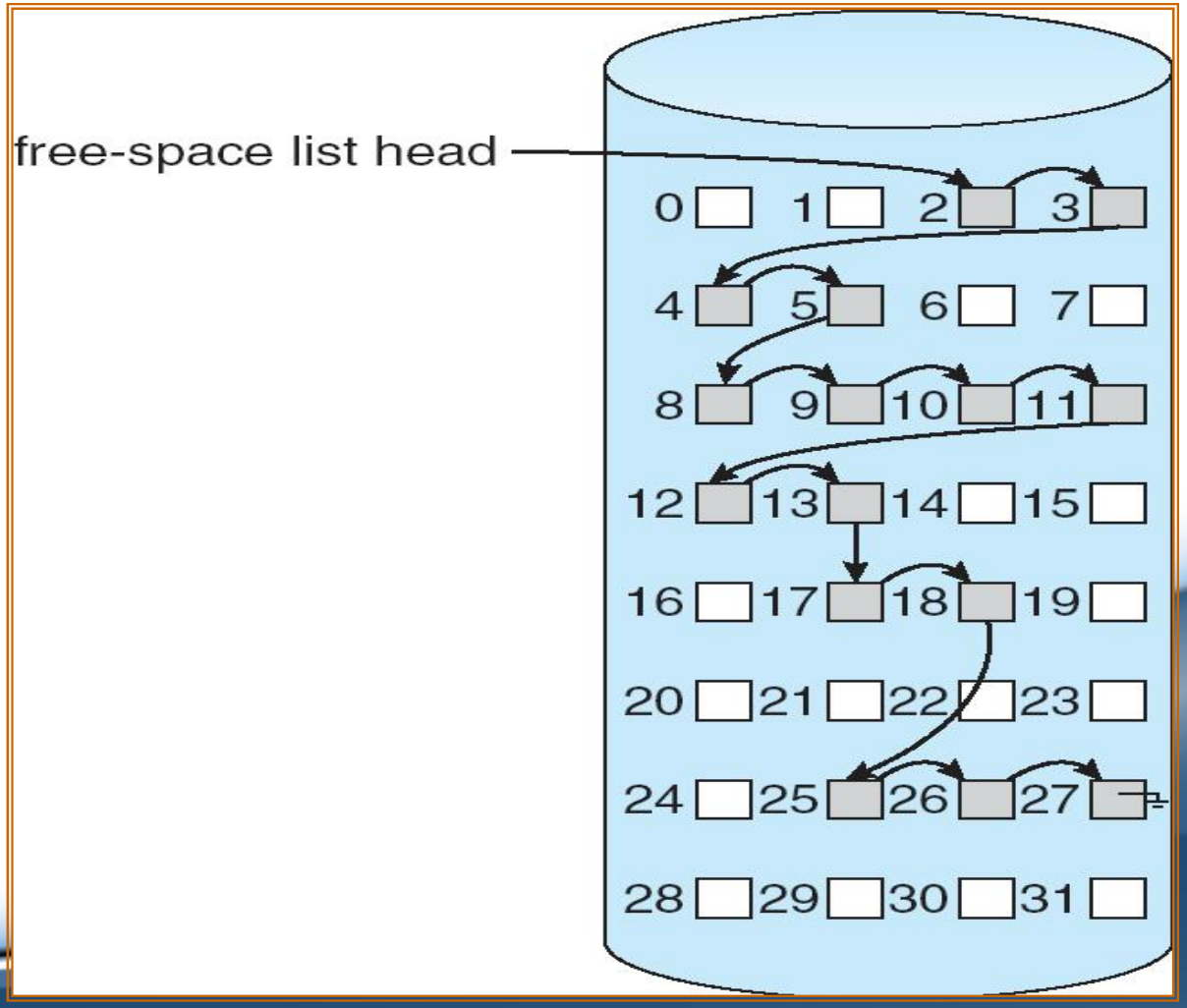
Bit Vector / Bit Map



Linked List

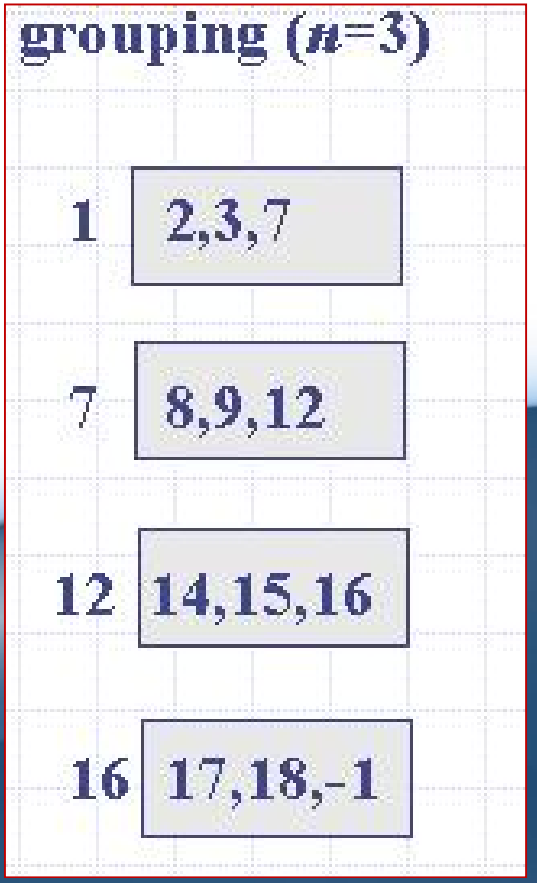
- Link together all the free disk blocks
- Keeps pointer to the first free block in a special location on the disk & cache it in the memory
- First block contains pointer to the next free disk block & so on
- Cannot get contiguous space easily
- No waste of space

Linked List


free-space list head
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31

Grouping

- Stores the address of n free blocks in the first free block.
- The first n-1 of these blocks are actually free
- The last contains the addresses of another n free blocks and so on



Counting

- Keeps the address of the first free block and the number n of free contiguous blocks that follow the first block.
- Each entry in the free space list consist of a disk address and a count.

counting: (1,3), (7, 3), (12, 1), (14, 5)

THANK YOU