Unit 6 – File Management

OS in File Manager

- The operating system is responsible for the following activities in connection with file management:
  - The creation and deletion of files.
  - The creation and deletion of directory.
  - The support of primitives for manipulating files and directories.
  - The mapping of files onto disk storage.
  - Backup of files on stable (non volatile) storage.

Introduction

- File is a collection of related information defined by user or creator
- Every file has a name and its data.
- Stored on the secondary storage device .i.e. disks or tape
- Commonly file represent programs (both source and object forms) and data
- Data files may be numeric, alphabetic or alphanumeric
- Files may be free-form, such as text files, or may be rigidly formatted
- Operating system associates various information with files, for example the date and time of the last modified file and the size of files etc.
- This information is called the file's attributes or metadata.
- The attributes varies considerably from system to system

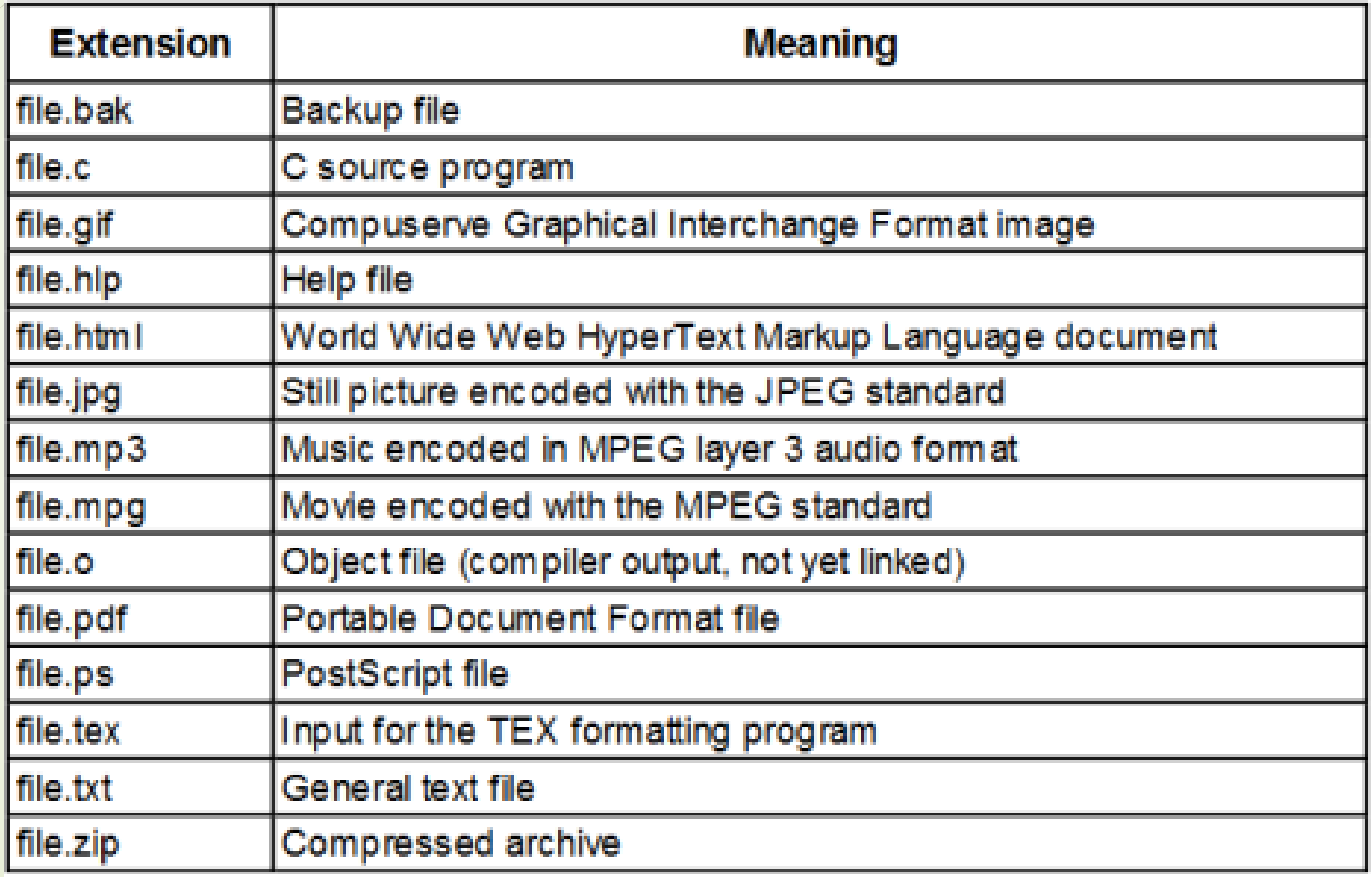
File Attributes

- **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** – It is a pointer to a device & to the location of the file on that 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
- **Read-only:** Allows a file to be read, but nothing can be written to the file.
- **Achieve:** Tells windows Backup to backup the file
- **System:** System file

File Naming

- When a process create a file, it gives the file name, while process terminates, the file continue to exist and can be accessed by other processes.
- A file is named, for the convenience of its human users, and it is referred to by its name.
- Support two part of names; separated by period; the part following period is called the file extension.
- Name of each file must be unique within the directory where it is stored.
- File names should be follow some naming convention.

File Naming



Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

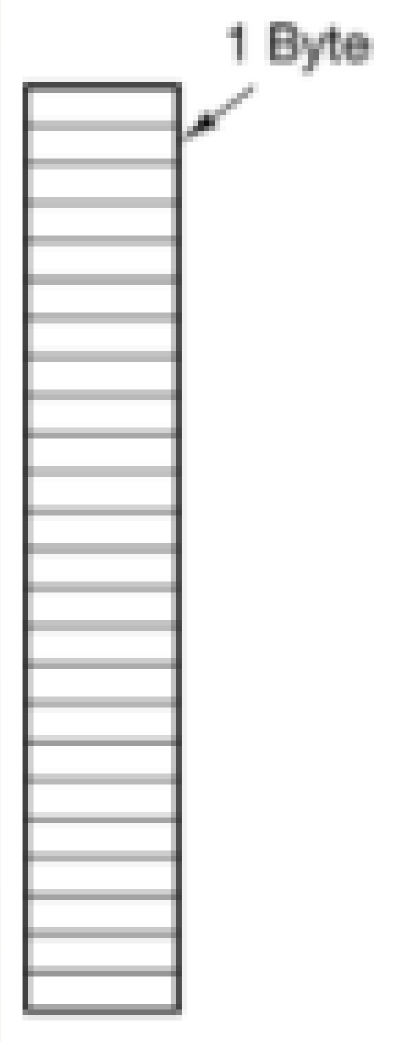
Fig: Some typical file extensions.

File Structure

- A file structure should be according to a required format that the OS understand.
- A has a certain structure according to its type.
- Three common possibilities or ways to struct files:
  - Unstructured
  - Record Structure
  - Tree Structure

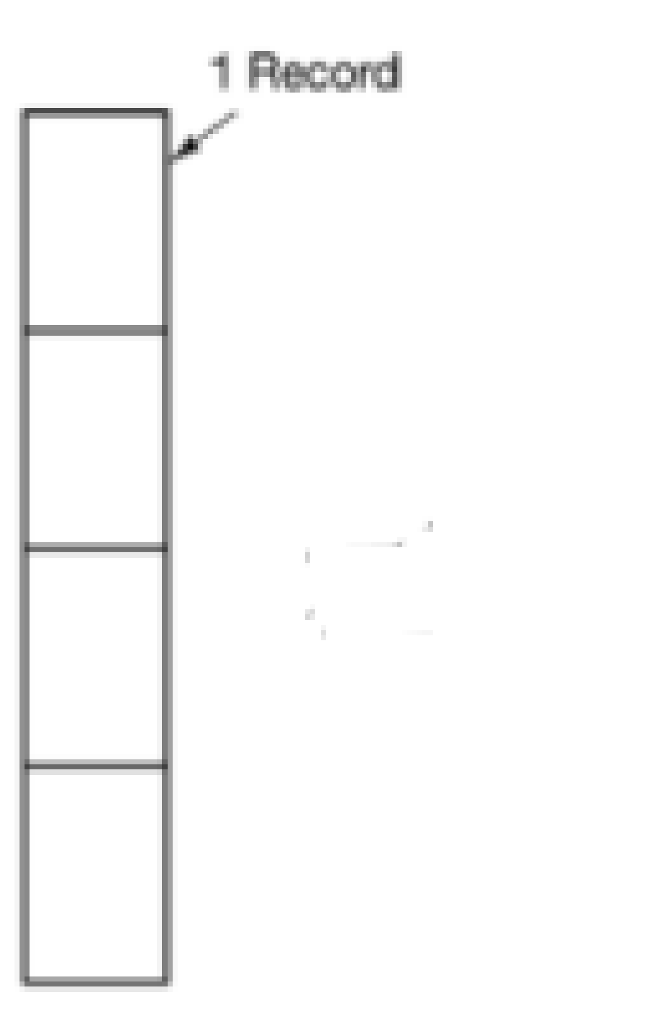
Unstructured File Structure

- Consist of unstructured sequence of bytes or words
- OS does not know or care what is in the file
- Any meaning must be imposed by user level programs
- Provides maximum flexibility
- User can put anything they want and name them, anyway that is convenient
- Both Unix and Windows use these approach



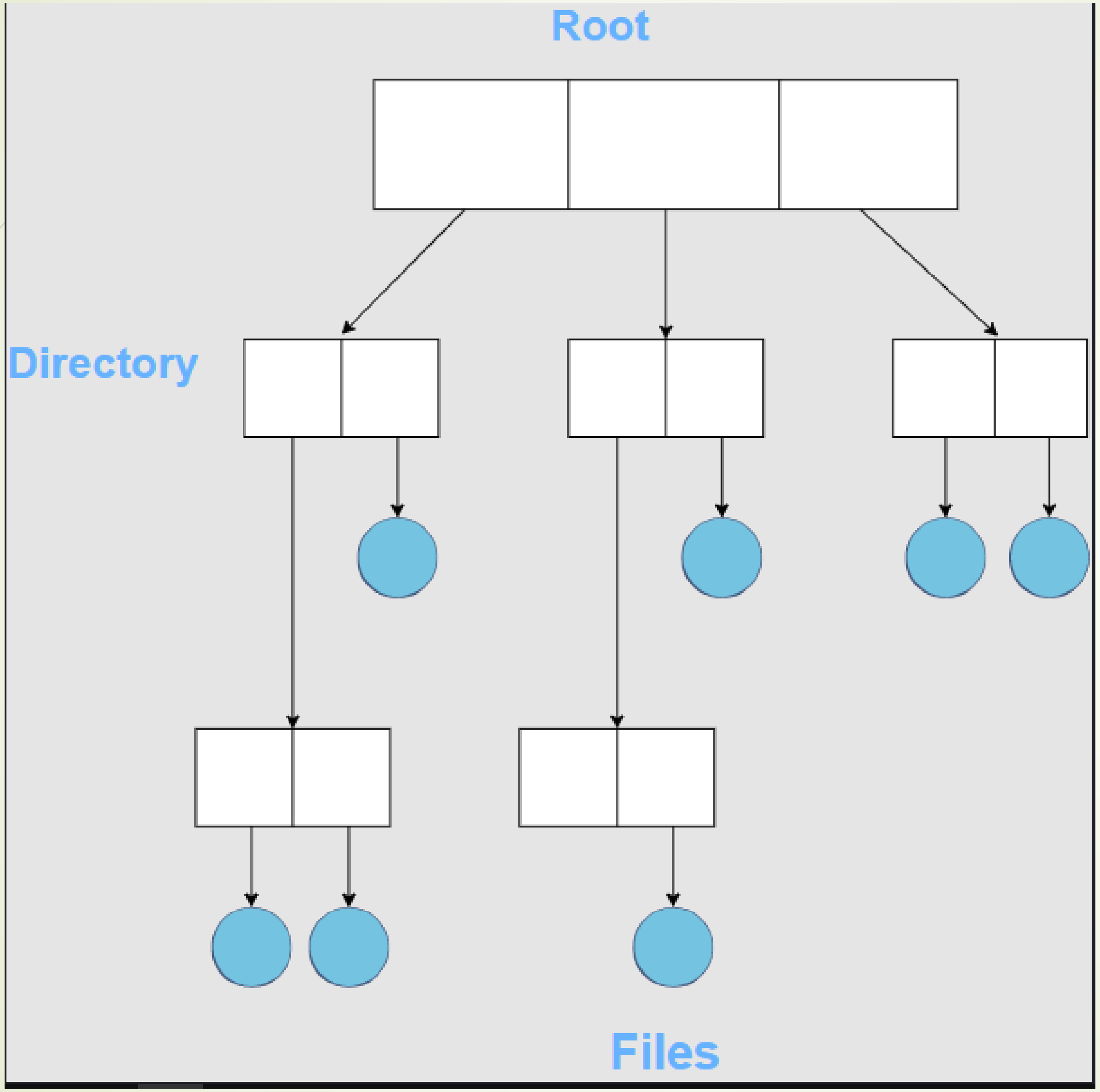
Record Structure

- A file is a sequence of fixed-length records, each with some internal structure
- Each read operation returns one records, and write operation overwrites or append one record
- Many old mainframe systems use this structure



Tree Structure

- File consists of tree of records, not necessarily all the same length
- Each containing a key field in a fixed position in the record, sorted on the key to allow the rapid searching
- The operation is to get the record with the specific key
- Used in large mainframe for commercial data processing
- This structure provides a clear parent-child relationship between directories.
- One example of a tree-based file system is the file system used in UNIX-like operating systems (e.g., ext4 or XFS).
- Root directory is the top-level directory.


Root
Directory
Files

File Types

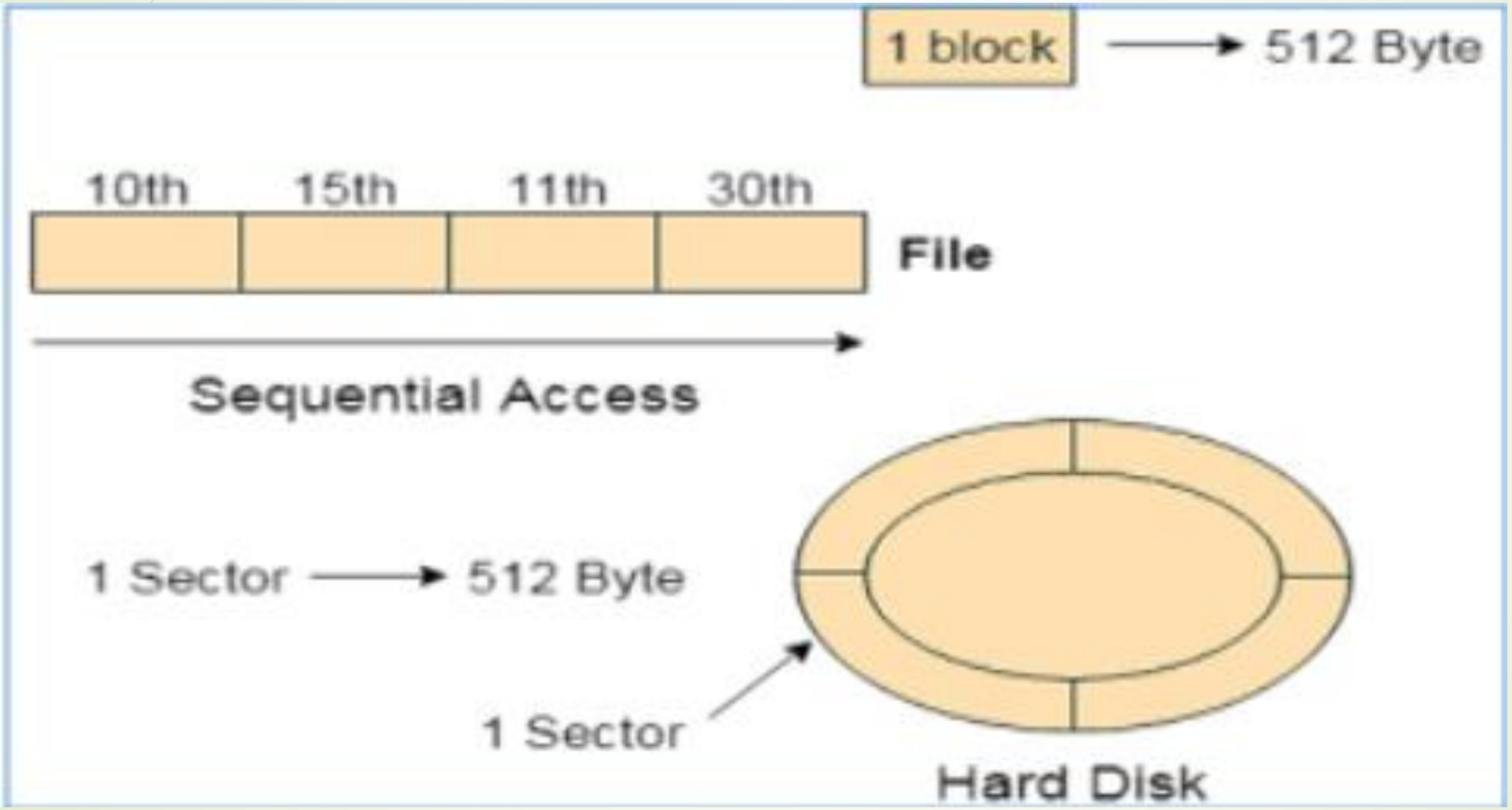
- **Regular Files**
  - Contains user information. Generally in ASCII or binary form
- **Directories**
  - System Files for maintaining the structure of file system
- **Character Special File**
  - Related to I/O of computer to model serial I/O device such as terminal
- **ASCII files**
  - Consist of line of text. They can be edited and printed with ordinary file editor
- **Binary Files**
  - Consist of sequence of bytes only

File Access

- **Sequential Access**

  - Early operating systems provided only one kind of file access
  - A process could read all the bytes or records in a file in order, starting at the beginning, but could not skip around and read them out of order
  - Sequential files could be rewound, however, so they could be read as often as needed
  - Sequential files were convenient when the storage medium was magnetic tape, rather than disk

File Access

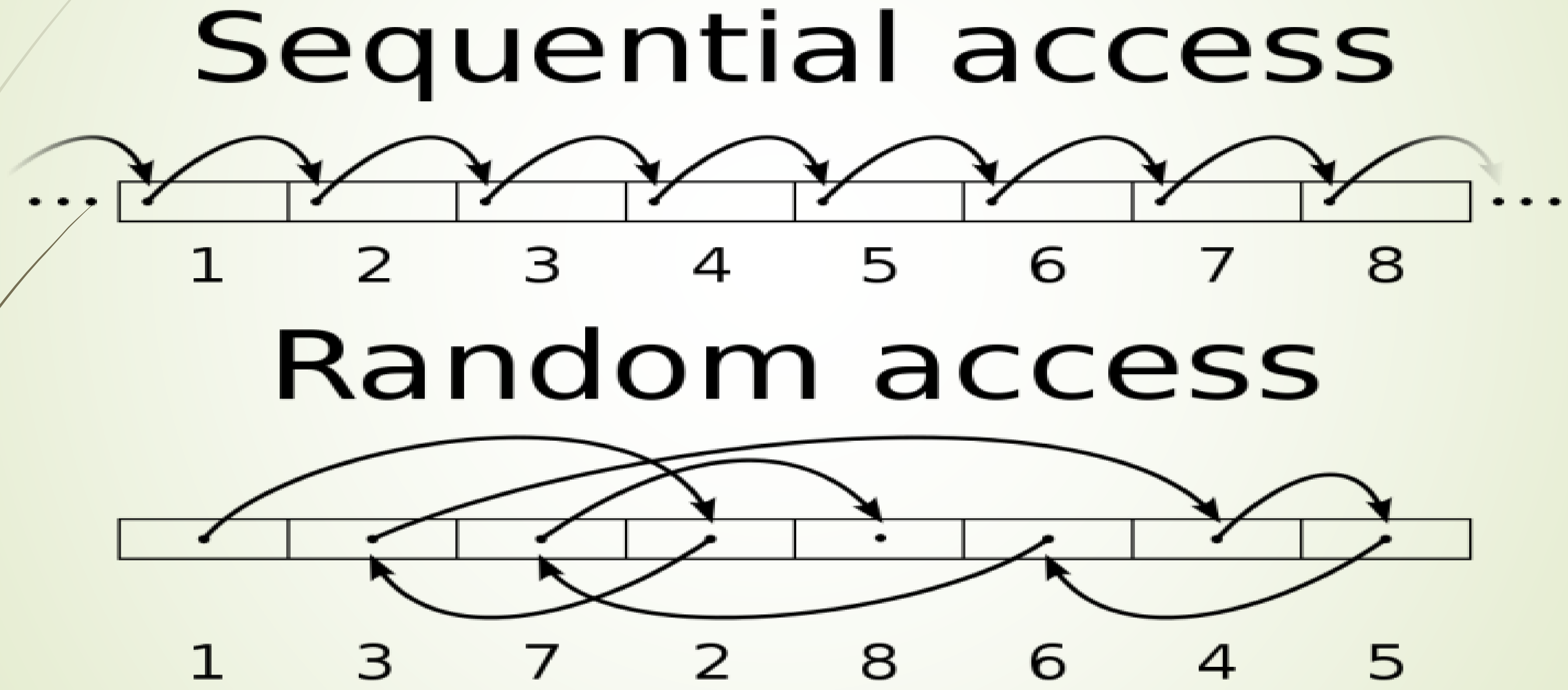

1 block
512 Byte
10th
15th
11th
30th
File
Sequential Access
1 Sector
512 Byte
1 Sector
Hard Disk

File Access (contd.)

- **Random / Direct Access File**
  - When disks came into use for storing files, it became possible to read the bytes or records of a file out of order, or to access records by key, rather than by position
  - Files whose bytes or records can be read in any order are called **random access files**
  - Essential for many applications, for example, database systems

- In some older mainframe operating systems, files are classified as being either sequential or random access at the time they are created.
- Modern operating systems do not make this distinction.
- All their files are automatically random access

File Access (contd.)



File Operations

- Create
- Delete
- Open
- Close
- Read
- Write
- Truncate
- Append
- Rename
- Seek
- Get Attribute
- Set Attribute

Directories

- Listing of the related information about files on the disk
- Each partition must have at least one directory.

- Single-Level Directory System
  - Also called root directory

- Two-level Directory System

- Hierarchical Directory System

File System Layout

- File system is a set of files, directories and other structures.
- Maintain information and identity where a file or directory’s data are located on a disk.
- All disk can be divided up into multiple partitions with independent file system on each partitions.
- Partition of disk sector, is called Master Boot Record(MBR), used to boot computer.

File System Layout

- **MRB:** Used to boot computer
- **Partition Table:** present at the end of the MRB.
  - Gives the start and end addresses of each partition.
- **Boot Block**
  - The boot block is available to start the operating system.
- **Superblock**
  - The super-block maintains information about the entire file system and includes the following fields:
    - Size of the file system
    - Number of data blocks in the file system
    - A flag indicating the state of the file system
    - Allocation group sizes

Implementing File

- The allocation method defines how the files are stored in the disk blocks.
- There are three main disk space allocation method

1. Contiguous
2. Linked Allocation
3. Indexed Allocation

Contiguous Allocation

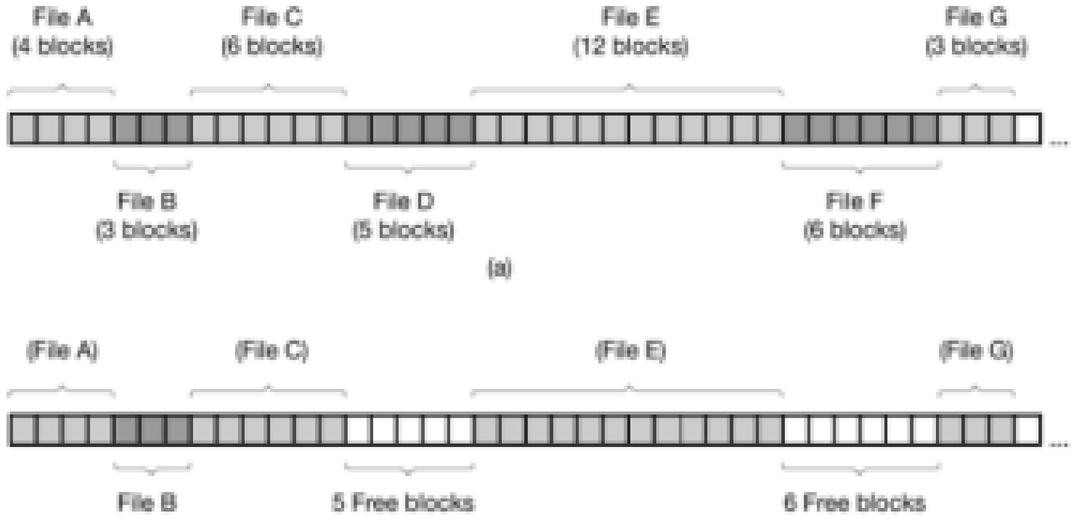


Fig: (a) Contiguous allocation of disk space for seven files. (b) The state of the disk after files *D* and *F* have been removed.

Linked List Allocation

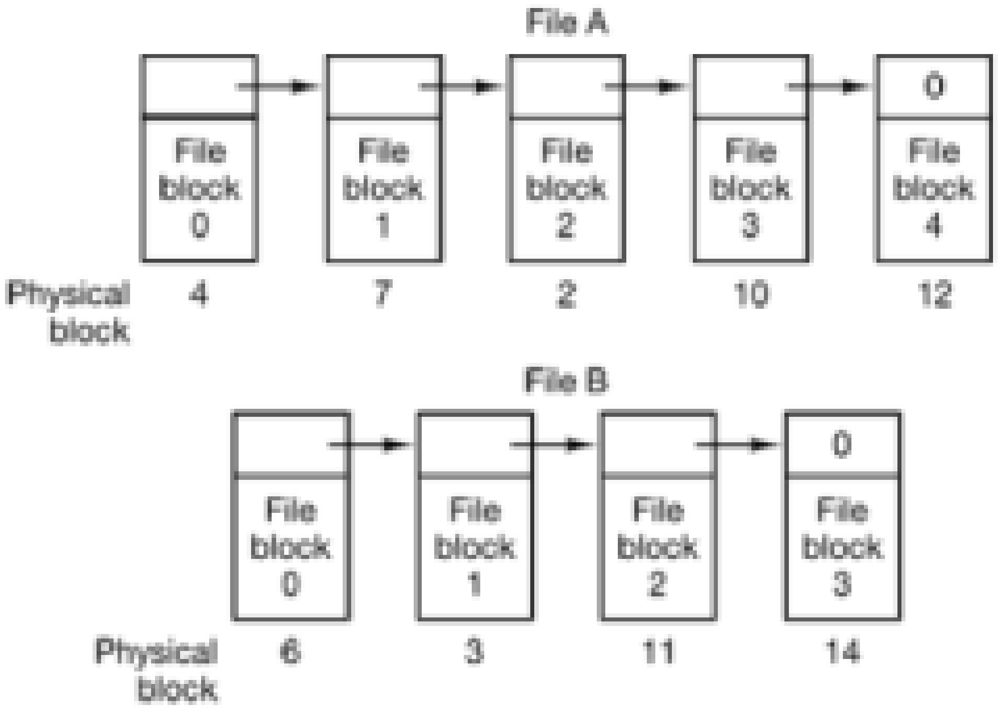
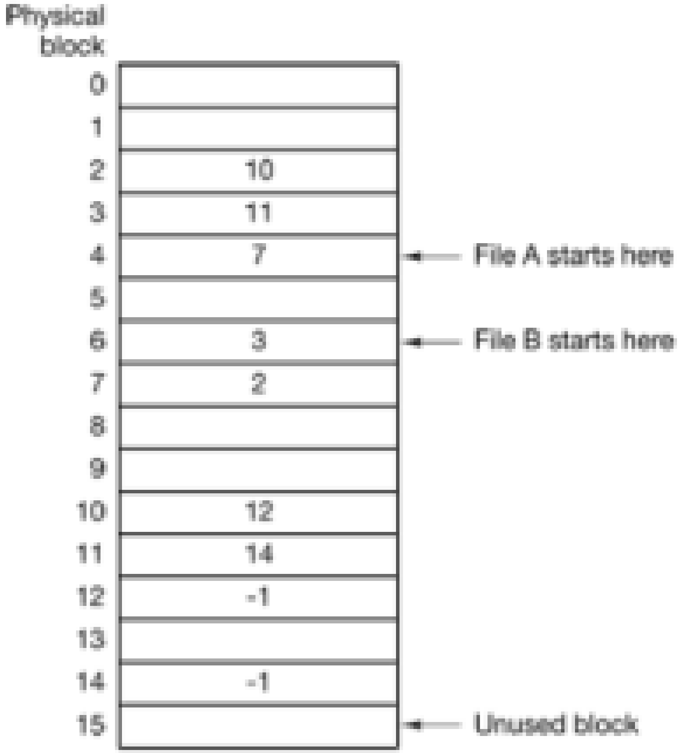


Fig: Storing a file as a linked list of disk blocks.

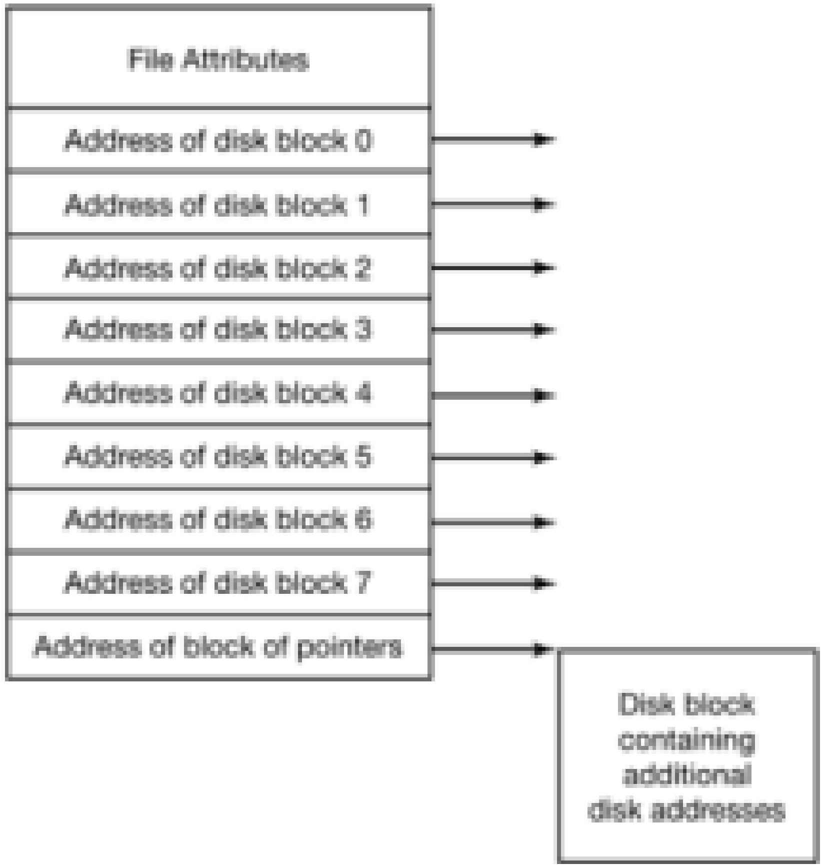
Linked List Allocation using FAT



Physical block	
0	
1	
2	10
3	11
4	7
5	
6	3
7	2
8	
9	
10	12
11	14
12	-1
13	
14	-1
15	

Fig: Linked list allocation using a file allocation table in main memory.

I-node Allocation



File Attributes

Address of disk block 0

Address of disk block 1

Address of disk block 2

Address of disk block 3

Address of disk block 4

Address of disk block 5

Address of disk block 6

Address of disk block 7

Address of block of pointers

Disk block containing additional disk addresses

Fig: An example of i-node

Directory Operations

- Create
- Delete
- Opendir
- Closedir
- Rename
- Link
- Unlink
- Readdir

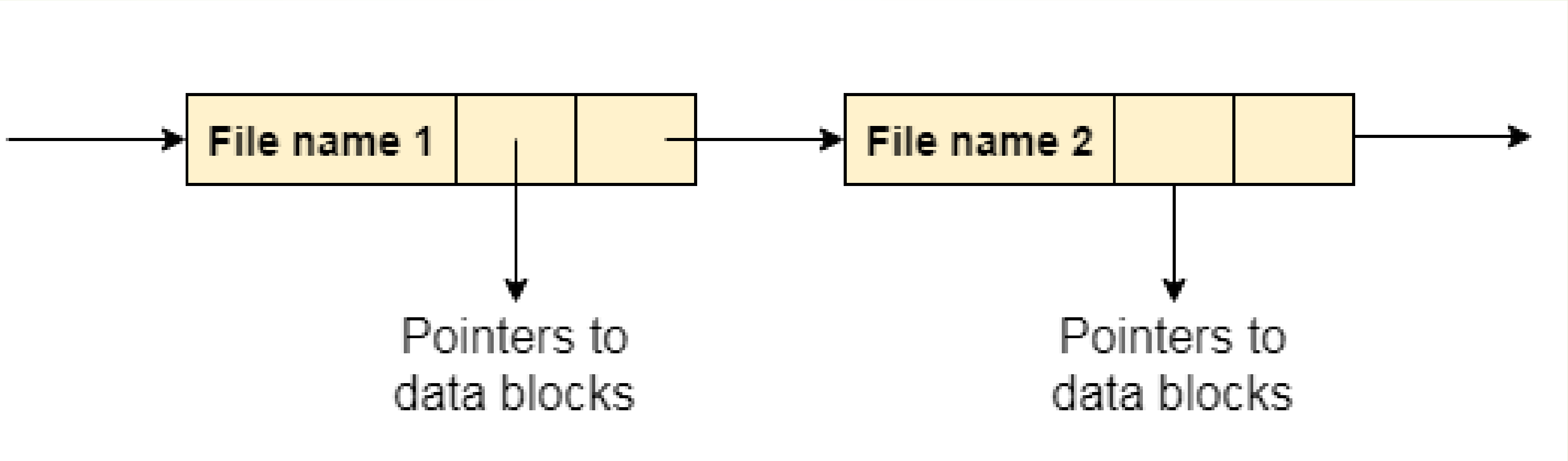
Path Names

- Absolute
  - Path name starting from root directory to the file
  - e.g. In Unix: /usr/user1/bin/lab2
  - E.g. In Windows: C:\Users\Ananta\Teaching_Materials\3rdSem\OS\Lecture_Slides
  - path separated by / in Unix and \ in windows
- Relative
  - A user can designate one directory as the current working directory
  - All path names not beginning at the root directory are taken relative to the working directory
  - E.g., bin/lab2 is enough to locate same file if current working directory is /usr/user1

Directory Implementation

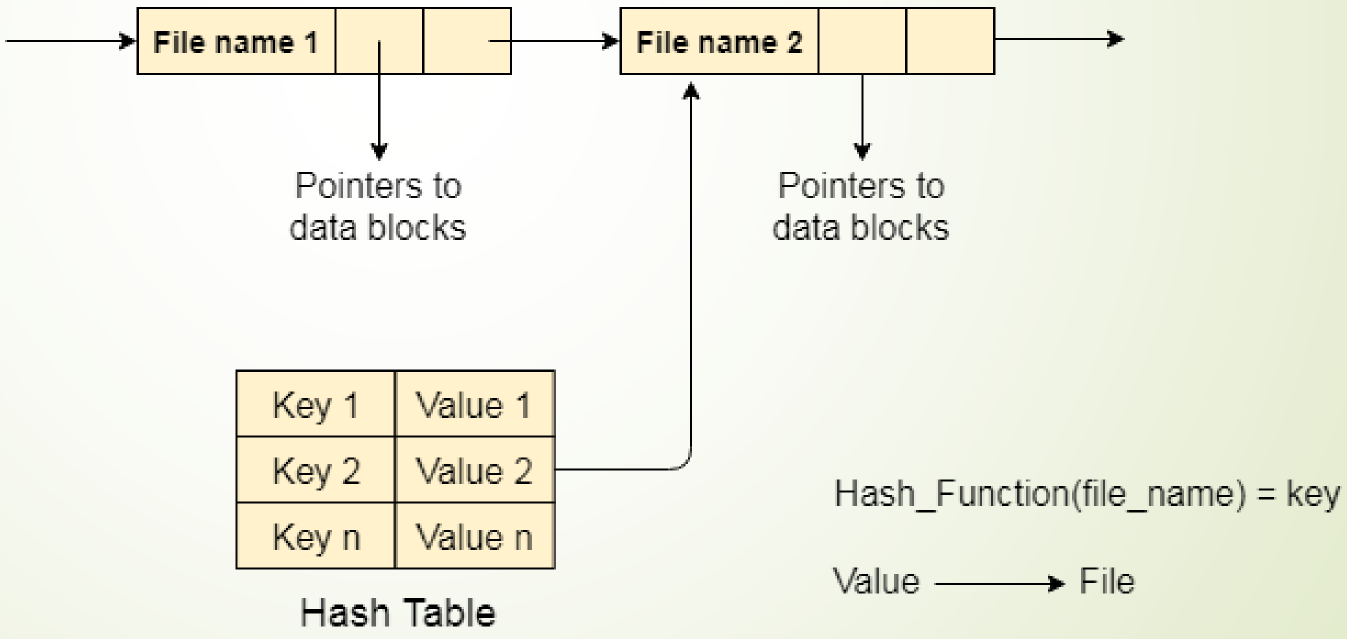
- Linear List
  - Use the linear list of the file names with pointer to the data blocks
  - It requires linear search to find a particular entry
  - Advantages: Simple to implement
  - Problem:
    - when a file is removed, a variable-sized gap is introduced
      - Next file to be entered may not fit
      - Searching time makes system inefficient

Linear List (contd.)



Directory Implementation (contd.)

- Hash Table
  - It consist linear list with hash table
  - Advantages: greatly decrease the file search time



Key 1	Value 1
Key 2	Value 2
Key n	Value n

Hash_Function(file_name) = key

Value → File

Shared Files

- When several users are working together on a project, they often need to share files

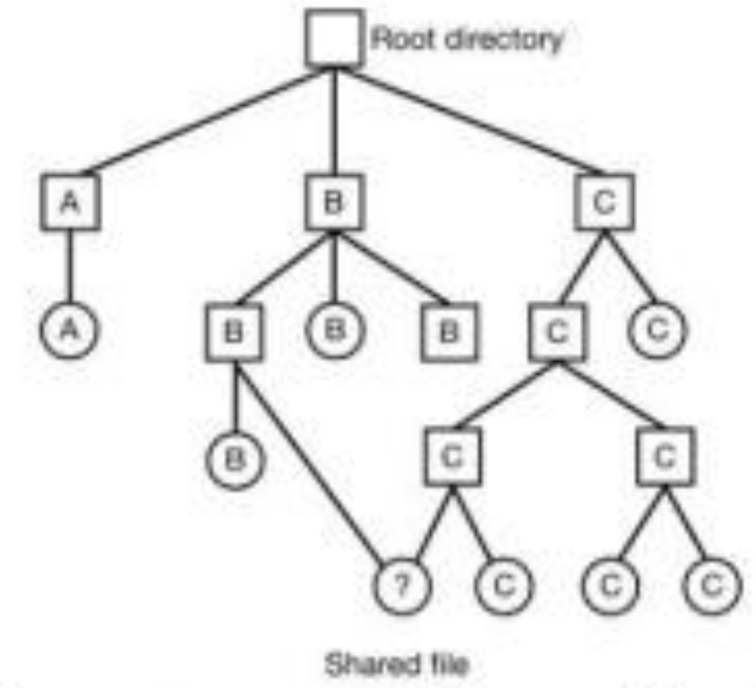


Fig: File system combining a shared file.

Free Space Management

- System keeps track of the free disk blocks for allocating space to files when they are created
- Also to reuse the space released from deleting the files
- The system maintains the free space lists
- Free space lists are implemented mainly by
  - Bitmaps and
  - Linked Lists

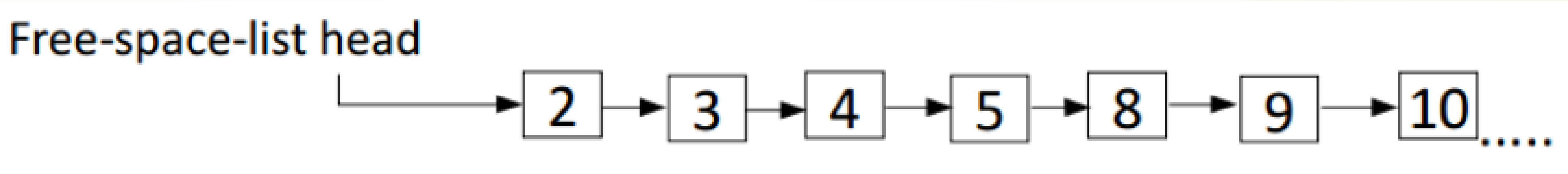
Bitmap or Bit-vector

- Each block is represented by a bit
- If the block is free, the bit is 1; if the block is allocated the bit is 0
- A disk with n blocks requires a bitmap with n bits
- Example: Consider a disk where blocks 2,3,4,5,8,9,10,11,12,13,17,18…are free, and rest are allocated

  The free space bit map would be
  0011110011111100010.........
- **Advantages**: Simple and efficient in finding first free block, or n consecutive free blocks
- **Problems**: Inefficient unless the entire bitmap is kept in main memory

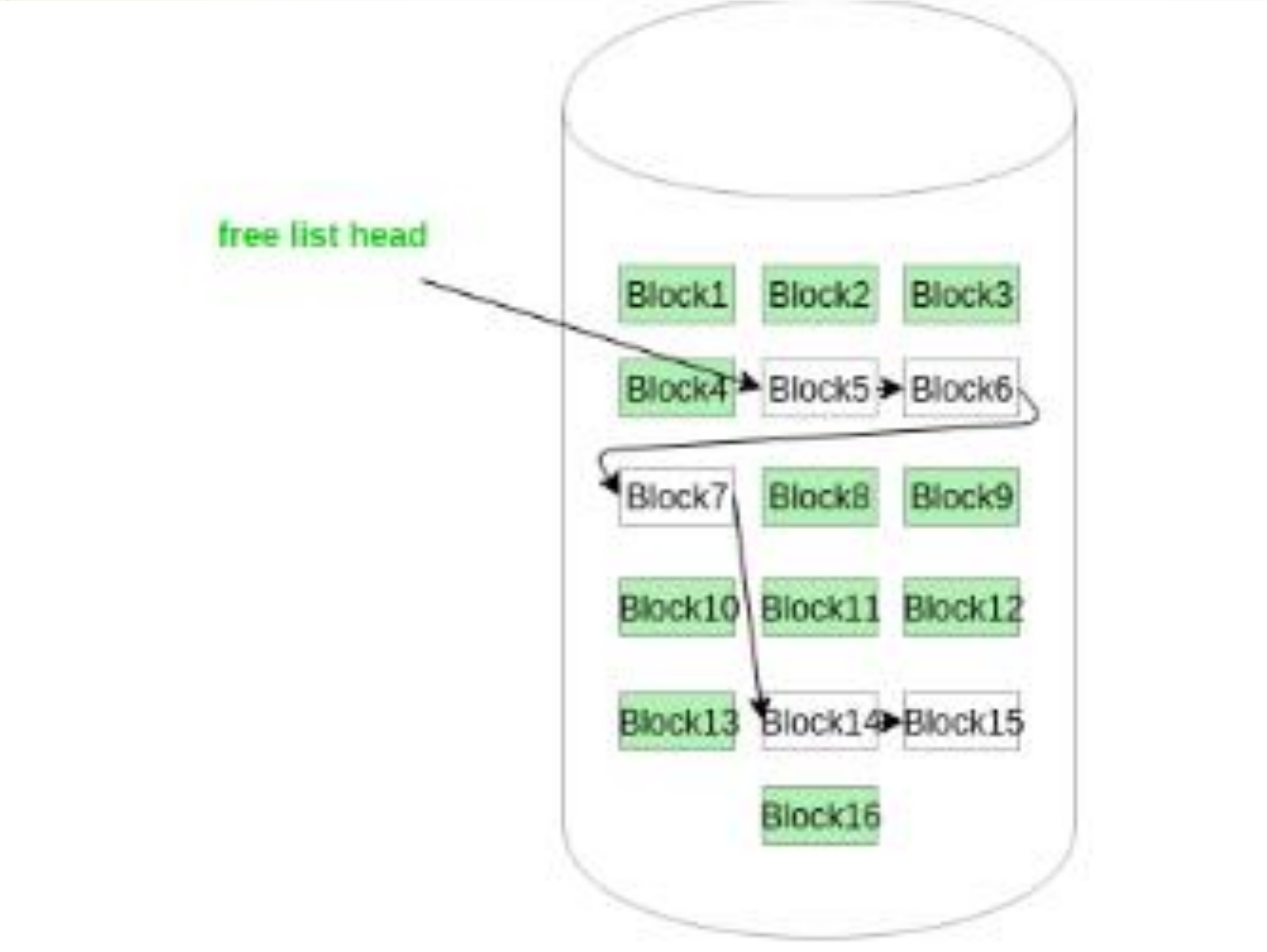
Linked List

- Free disk blocks are linked together
- Keeping a pointer to the first free block in a special location on the disk and caching it in memory
- The first block contains a pointer to the next free block



- Advantages: Only one block is kept in memory
- Problems: Not efficient; to traverse list, it must read each block

Linked List


free list head
Block1
Block2
Block3
Block4
Block5
Block6
Block7
Block8
Block9
Block10
Block11
Block12
Block13
Block14
Block15
Block16