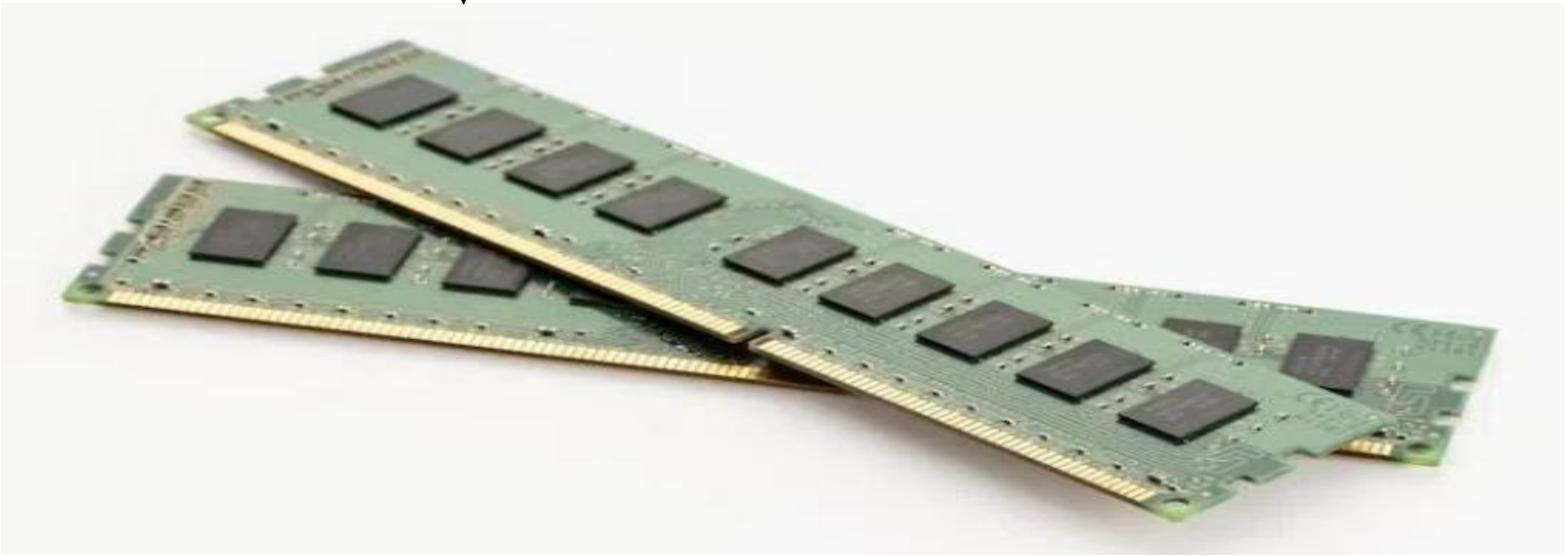
Operating System

Unit – 6
(Part - A)
File Management

Mayank Mishra
School of Electronics Engineering
KIIT-Deemed to be University

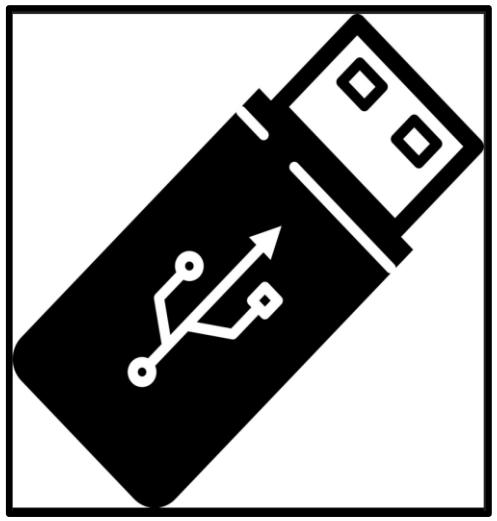
Storage Management

- Program has to be loaded to main memory for the execution.
- But **main memory** is usually too small to accommodate all the data and programs permanently.



Storage Management (Contd.)

- So, the computer must provide secondary storage to back up main memory.

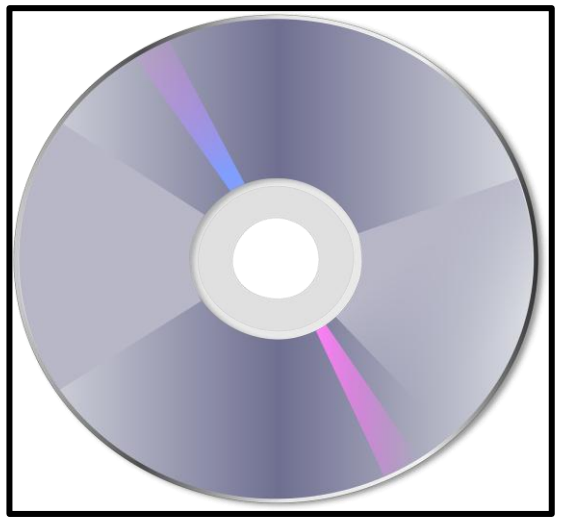


Figure Source: https://www.vectorstock.com/ , https://pixabay.com/

- Most computers use disk as the most common secondary storage device for both programs and data.

- The FILE SYSTEM provides the mechanism for:
  - Storage of data and programs on the disk.
  - Access to data and programs on the disk

- **Files** are mapped by OS to physical devices.

A collection of related information defined by its creator

- Files are normally organized into directories for ease of use.

Figure Source: https://www.freepik.com/

- Different storage devices vary in many aspects.

  E.g.

  - Some device transfer a character or a block of character at a time.
  - Some device can be accessed only sequentially.
  - Some can be accessed randomly.
  - Some transfer data synchronously.
  - Some transfer data asynchronously.
  - Some are dedicated while some are shared.
  - Some can be read only.
  - Some can be read-write

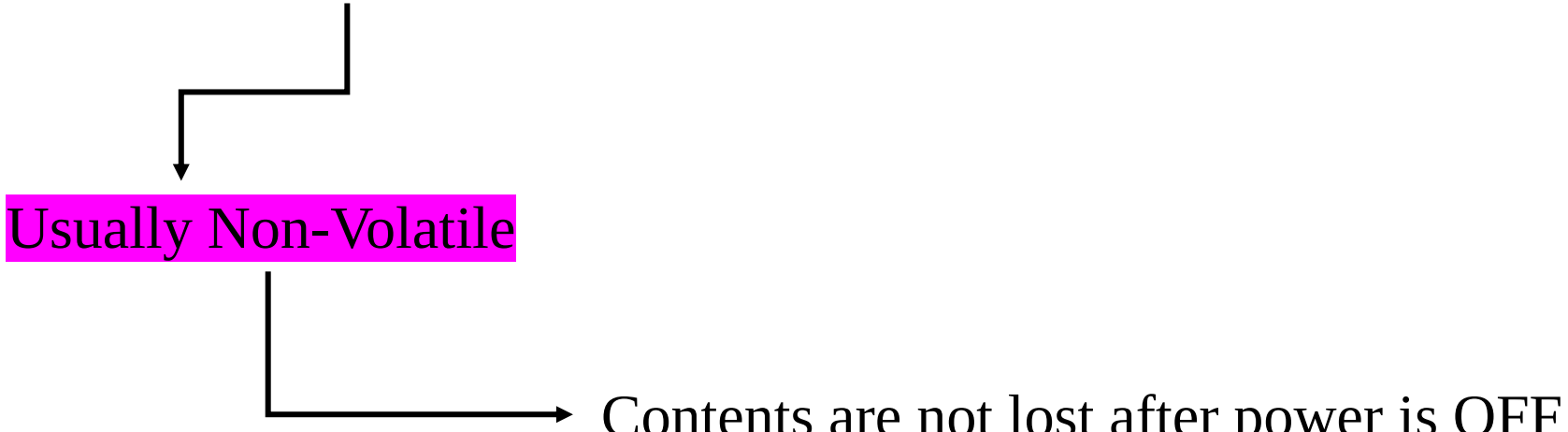
Because of all this device variation, the operating system needs to provide a wide range of functionality to applications, to allow them to control all aspects of the devices.

Concept of File

- Different storage media can be used for storing information in a computer (Example: Magnetic disks, magnetic tapes, and optical disks, etc.)

- The operating system abstracts from the physical properties of its storage devices to define a logical storage unit- **The File**.

- Files are mapped by the OS onto physical devices.

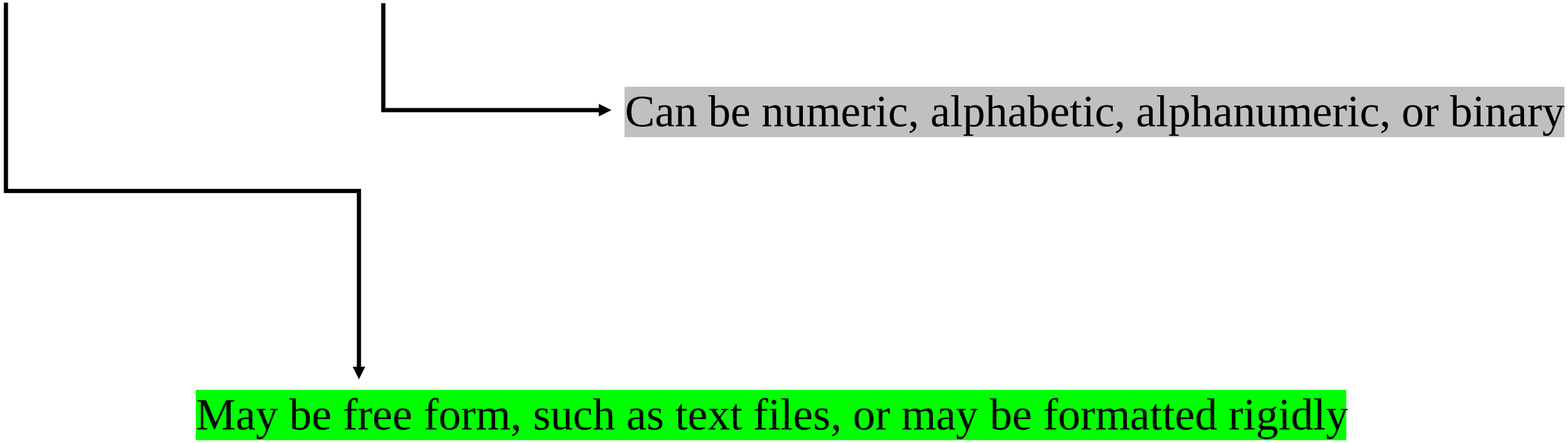


Concept of File (Contd.)

- A file is a named collection of related information that is recorded on secondary storage.

- From a User's perspective:
  - A file is the smallest allotment of logical secondary storage.
  - Data cannot be written to secondary storage unless they are within a file.

- Files represents both data and programs.



In general, a file is a sequence of bits, bytes, lines, or records, the meaning of which is defined by the file's creator and user.

- The information in a file is defined by its creator.

- **Many different types of information may be stores in a file. *For example:***

Source Programs

Object Programs

Executable Programs

Numeric Data

Text

Graphic Images

Sound Recordings

Video Recordings

File Attributes

- A file is named, for the convenience of its human users, and is referred to by its name. A name is usually a string of characters, such as ***example.c*** .

- Some systems differentiate between uppercase and lowercase characters in names, whereas other systems do not.

- When a file is named, it becomes independent of the process, the user, and even the system that created it.
  - For instance, one user might create the file **example.c**, and another user might edit that file by specifying its name. The file's owner might write the file to a USB drive, send it as an e-mail attachment, or copy it across a network, and it could still be called example.c on the destination system.

A file's attributes vary from one operating system to another but typically consist of these:

- **Name:** The symbolic file name is the only information kept in human readable form.
- **Identifier:** This unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file.
- **Type:** This information is needed for systems that support different types of files.
- **Location:** This information is a pointer to a device and to the location of the file on that device.
- **Size:** The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.
- **Protection:** Access-control information determines who can do reading, writing, executing, and so on.
- **Timestamps and user identification:** This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

File Operations

- The OS has to provide **system calls** to perform various file operations.
- We will see what OS actually does in order to perform these operations.

Creating a File

- Two steps are necessary to create a file.
- First, space in the file system must be found for the file.
- Second, an entry for the new file must be made in a directory.

File Operations (Contd.)

Writing a File

- Make a system call specifying both the name of the file and the information to be written to the file.
- With the given name of the file, the system searches for the file in the directory and locates it.
- A write pointer is maintained which points to the location in the file from where the next write must take place.
- After the writing is done, the write pointer is updated.

File Operations (Contd.)

Reading a File

- Make a system call specifying both the name of the file and where (in memory) the next block of the file should be read.
- With the given name of the file, the system searches for the file in the directory and locates it.
- A read pointer is maintained which points to the location in the file from where the next read must take place.
- After the reading is done, the read pointer is updated.
- Mostly read and write operation use the same pointer.

File Operations (Contd.)

Repositioning within a file.

- The directory is searched for the appropriate file.
- The current-file-position pointer is repositioned to a given value.
- Reposition within a file need not to involve any actual I/O.

File Operations (Contd.)

Deleting a file.

- The directory is searched for the appropriate file.
- The current-file-position pointer is repositioned to a given value.
- Reposition within a file need not to involve any actual I/O.

File Operations (Contd.)

Truncating a file.

- The user may want to erase the contents of a file but keep its attributes.
- Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged—except for file length.
- The file can then be reset to length zero, and its file space can be released.

File Operations (Contd.)

Note:

- Most of the common file operations that we discussed involve searching the directory for the entry associated with the named file.
- To **avoid this constant searching**, many systems require that an **open( ) system call** be made before a file is first used actively.
- The OS keeps a **small table**, called the **open-file 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 the file is no longer being actively used, it is closed by the process, and the operating system removes its entry from the **open-file table.**

Access Methods

- The information in files can be accessed in several ways. Some systems provide just one access method for files while other systems may support many access methods.

- Two most common access methods are:
  1. Sequential Access
  2. Direct Access

Access Methods (Contd.)

Sequential Access

- The simplest access method is sequential access.
- Information in the file is processed in order, one record after the other.
- This mode of access is by far the most common; for example, editors and compilers usually access files in this fashion.

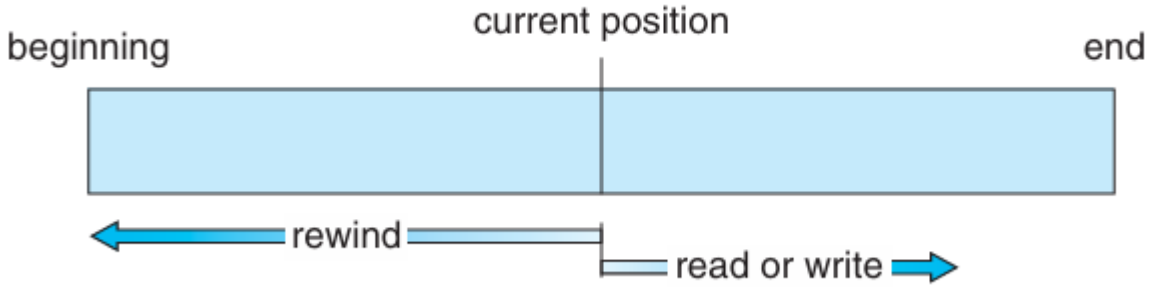


Fig. Sequential-access file.

Access Methods (Contd.)

Sequential Access

- Reads and writes make up the bulk of the operations on a file. A read operation—**read next()**—reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location.

- Similarly, the write operation—write next()—appends to the end of the file and advances to the end of the newly written material (the new end of file).

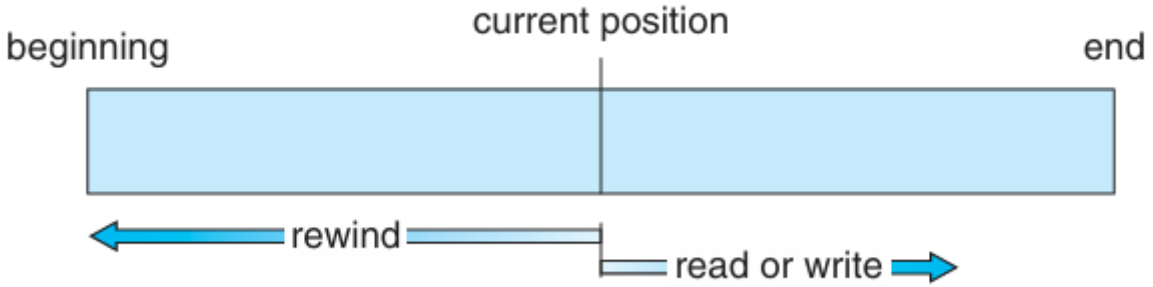


Fig. Sequential-access file.

Access Methods (Contd.)

Direct Access (or Relative Access)

- Here, a file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
- For direct access, the file is viewed as a numbered sequence of blocks or records.
- Thus, we may read block 14, then read block 53, and then write block 7. There are no restrictions on the order of reading or writing for a direct-access file.
- Direct-access files are of great use for immediate access to large amounts of information. Example – Databases.

Access Methods (Contd.)

Direct Access (or Relative Access)

- For the direct-access method, the file operations must be modified to include the block number as a parameter.

- Thus, we have
  - **read(n) –** reads from the block number ‘n’ from the files.
  - **write(n) –** writes the block ‘n’ of the files.

- The block number provided by the user to the operating system is normally a relative block number. A relative block number is an index relative to the beginning of the file.

Directory Structure

- The file systems of computers can be extensive. Systems need to store huge number of files on the disks.

- So, to manage this huge amount of data, they must be organized properly.

Directories are used for organizing this data.

Figure Source: https://www.freepik.com/

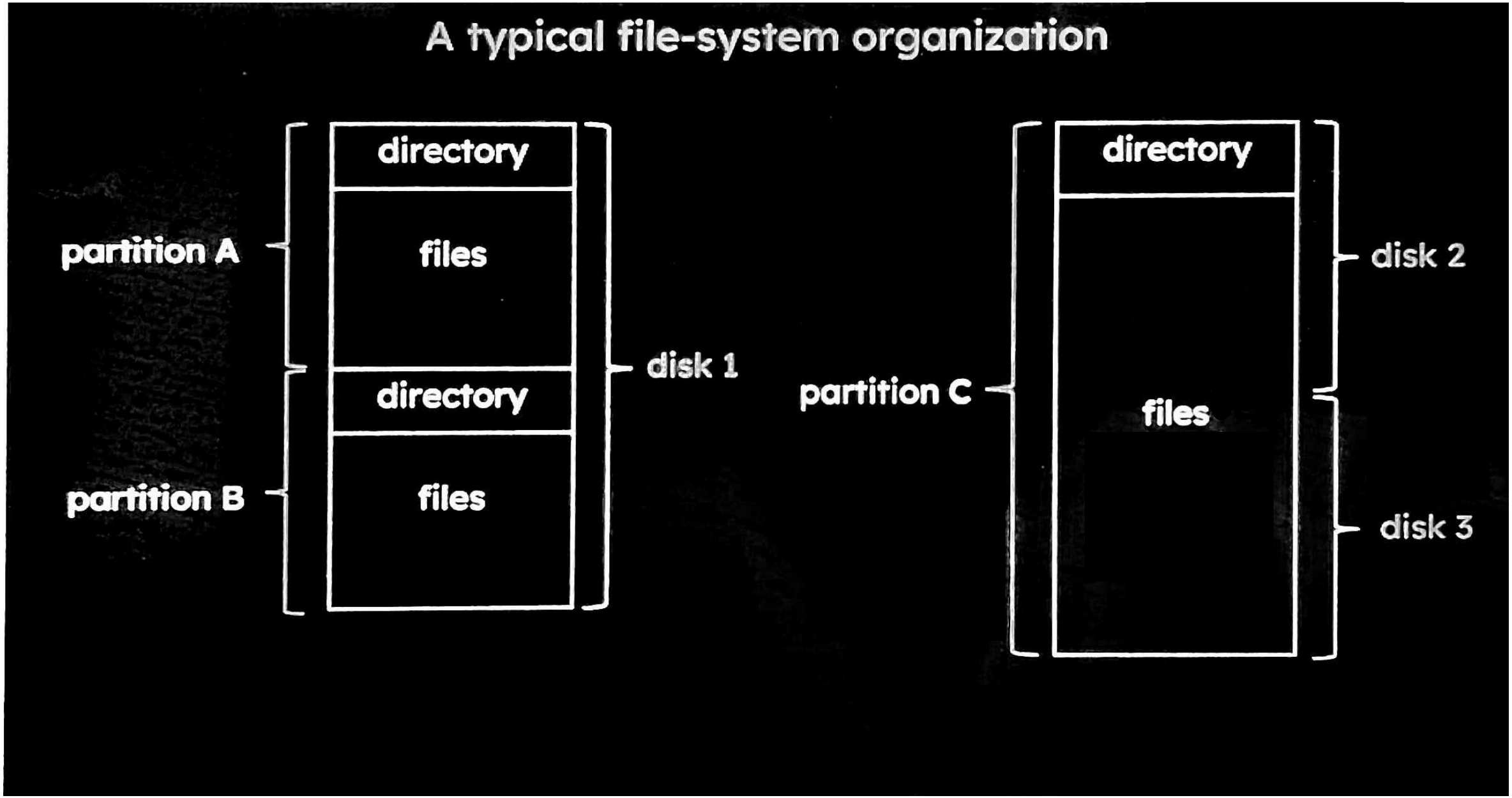
Storage Structure

(Before going to Directory Structure lets have an idea of storage structure, so that we can know where the directory lies.)

- A disk or any large storage device can be used entirely for a file system.
- But it is better to have multiple file systems on a disk or use a parts of a disk for a file system and other parts for other things like swap space, raw disk space, etc.
- These parts are called **Partitions** or **Slices** or **Mini-disk.**
- A file system can be created on each of these parts of the disk.
- These parts can also be further combined to form larger structures called **Volumes**.
- A file system can be created on these Volumes as well.

Storage Structure

- Each Volume can be thought of as a virtual disk.
- Volumes can store multiple Operating Systems allowing the system to boot and run more than one Operating System.
- Each volume that contains file system must also contain information about the files in the system.
  - This information is kept in entries in a device directory
- A directory records information such as ***name, location, size, type*** for all files on that volume.


A typical file-system organization
directory
files
partition A
directory
files
partition B
disk 1
directory
files
partition C
disk 2
disk 3

Directory Overview

- Operations that can be performed on a directory:

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

- Following are the most common schemes for defining the logical structure of a directory.

  - Single-Level Directory
  - Two-Level Directory
  - Tree-Structured Directories
  - Acyclic-Graph Directories
  - General Graph Directory

Note: This is a part of Assignement-2

File Allocation Methods

- The allocation methods define how the files are stored in the disk blocks. There are three main file allocation methods.

  - Contiguous Allocation
  - Linked Allocation
  - Indexed Allocation

- The main idea behind these methods is to provide:

  - Efficient disk space utilization.
  - Fast access to the file blocks.

Contiguous Allocation

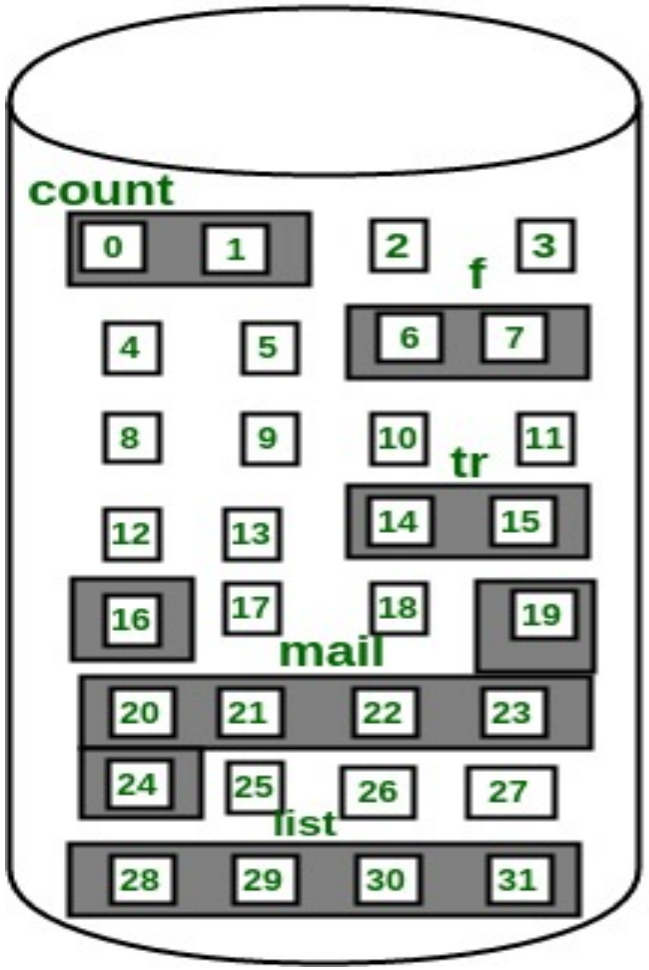
- In this scheme, each file occupies a contiguous set of blocks on the disk.

- **For example**, if a file requires **n** blocks and is given a block **b** as the **starting location**, then the blocks assigned to the file will be: b, b+1, b+2,……b+n-1. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.

- The directory entry for a file with contiguous allocation contains:

  - Address of starting block
  - Length of the allocated portion.

Contiguous Allocation (Contd.)



Directory

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

The *file 'mail'* in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies *19*, *20*, *21*, *22*, *23*, *24* blocks.

Contiguous Allocation (Contd.)

- **Advantages:**
  - Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).
  - This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

- **Disadvantages:**
  - This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.

  - Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

Contiguous Allocation

- In this scheme, each file occupies a contiguous set of blocks on the disk.

- **For example**, if a file requires **n** blocks and is given a block **b** as the **starting location**, then the blocks assigned to the file will be: b, b+1, b+2,……b+n-1. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.

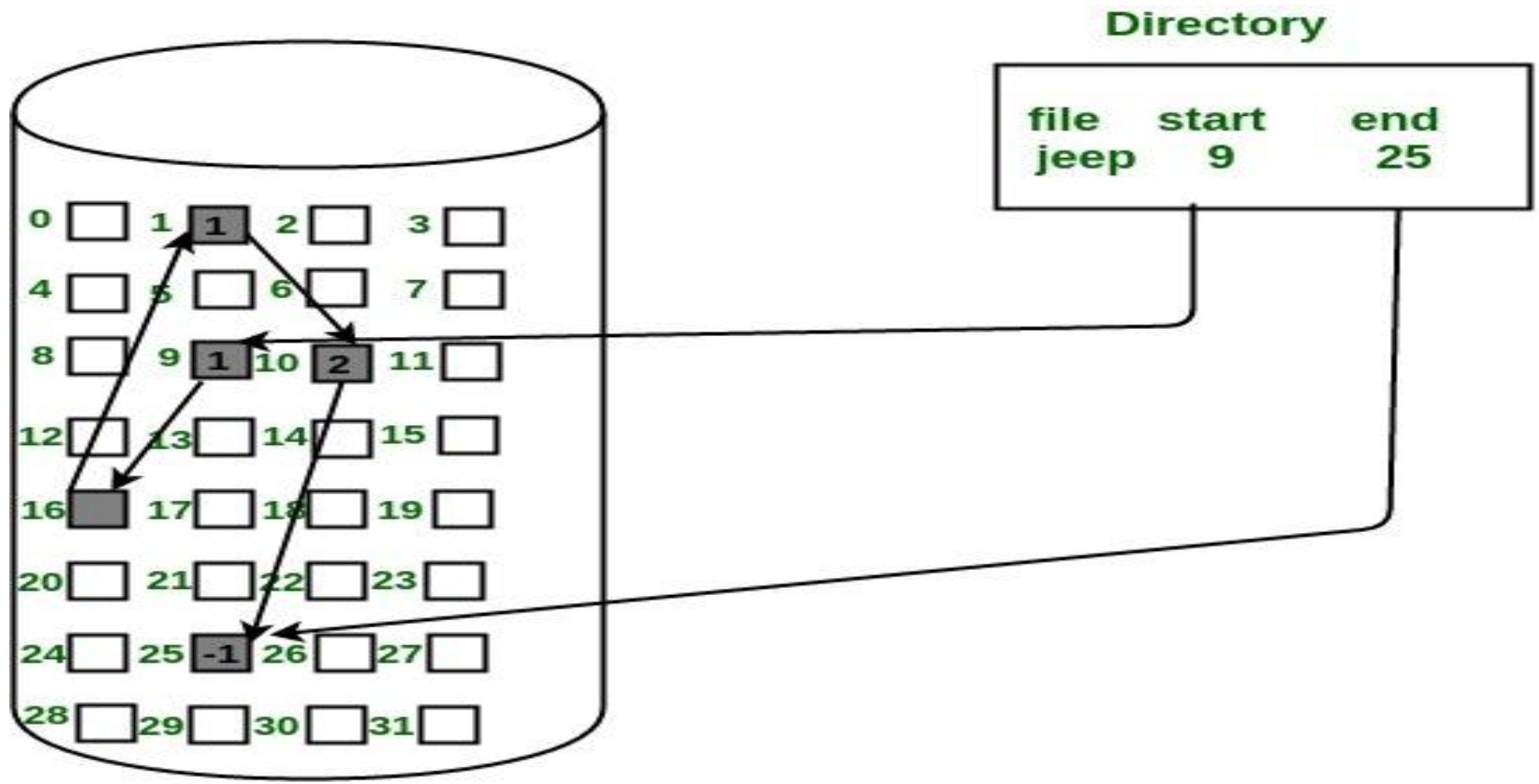
- The directory entry for a file with contiguous allocation contains:

  - Address of starting block
  - Length of the allocated portion.

Linked List Allocation

- In this scheme, each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk.

- The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.


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

Linked List Allocation (Contd.)

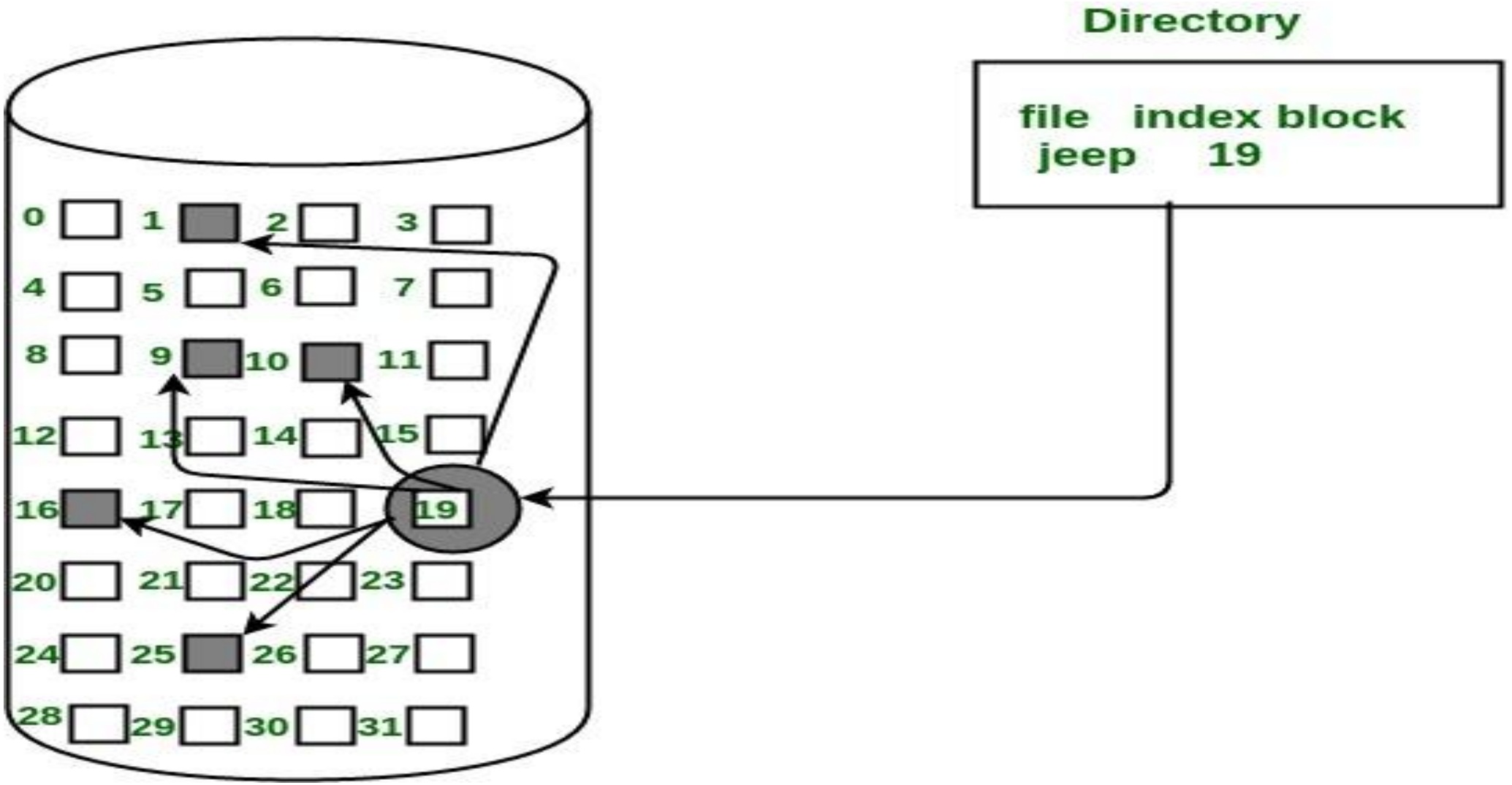
- **Advantages:**
  - This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
  - This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.
- **Disadvantages:**
  - Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.
  - It does not support random or direct access. We can not directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access) from the starting block of the file via block pointers.
  - Pointers required in the linked allocation incur some extra overhead.

Indexed Allocation

- In this scheme, a special block known as the **Index block** contains the pointers to all the blocks occupied by a file.

- Each file has its own index block. The ith entry in the index block contains the disk address of the ith file block.

The directory entry contains the address of the index block as shown in the image:



Indexed Allocation(Contd.)

- **Advantages:**
  - This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
  - It overcomes the problem of external fragmentation.
- **Disadvantages:**
  - The pointer overhead for indexed allocation is greater than linked allocation.
  - For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.

References

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, “Operating System Concepts,” Eleventh Edition (Willey).

2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition (Pearson Publications), 2014.

3. https://www.geeksforgeeks.org/

4. https://www.javatpoint.com/

5. https://www.tutorialspoint.com/

6. https://www.nesoacademy.org/

7. https://www.tpointtech.com/