Operating Systems

Course Code: **71203002004**

File Organization

by -

Minal Rajwar

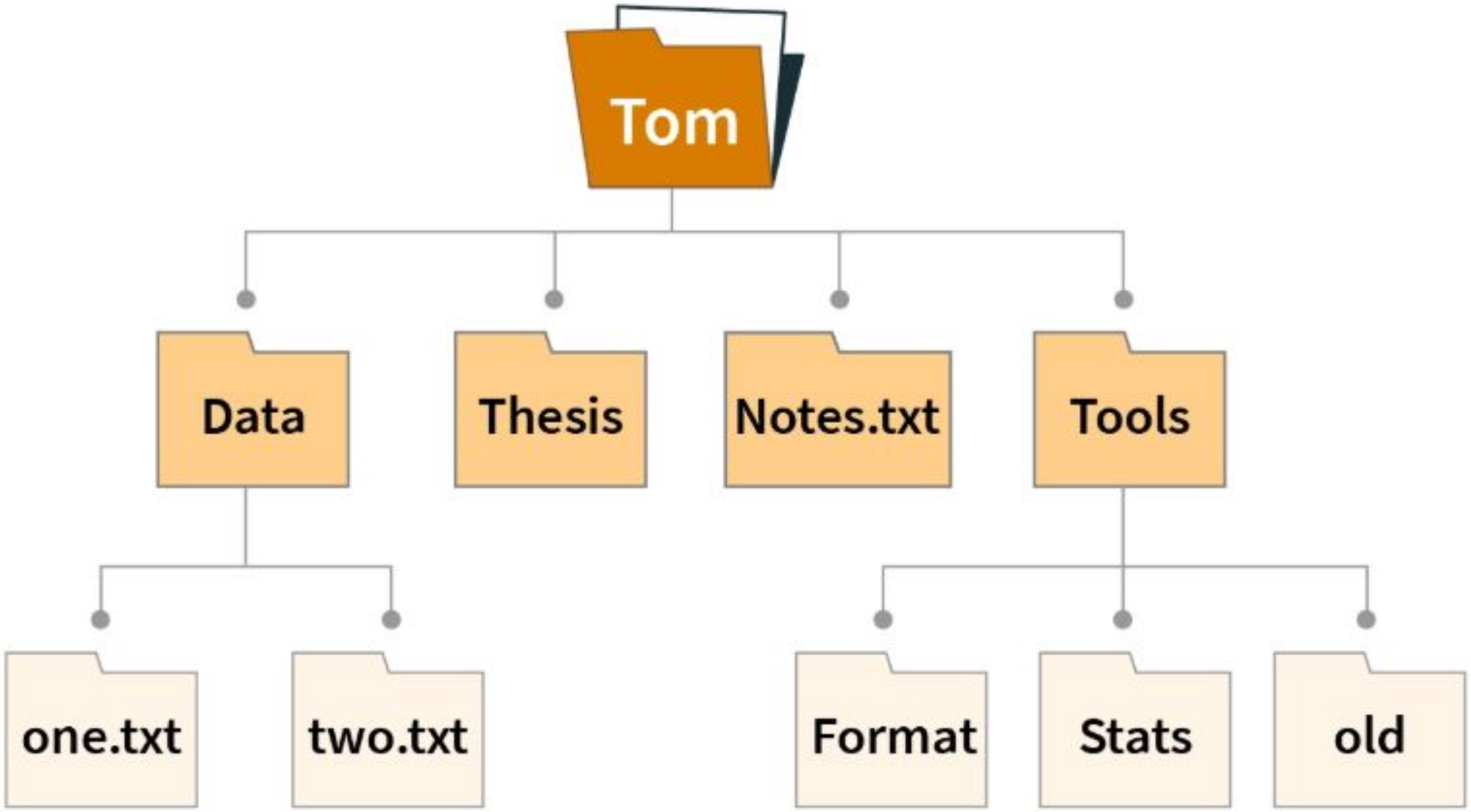
File Organization

File organization in an operating system determines how data is stored and arranged on a storage device, such as a hard drive, flash drive, or other memory.

Storage devices are often divided into partitions, and each partition is formatted with a filesystem.

The filesystem breaks the data into manageable pieces called **files** and **directories**.

It also keeps information about the files, like their name, type, and access permissions.


Tom
Data
Thesis
Notes.txt
Tools
one.txt
two.txt
Format
Stats
old

Advantages of Using a File System

1. **Organized Storage:** Keeps data structured, making it easier to find and access files.
2. **Efficient Retrieval:** Allows fast access to files using directories and indexing.
3. **Security and Permissions:** Controls who can view or modify files through access rights.
4. **Data Integrity:** Protects files from corruption using features like journaling and checksums.
5. **Space Management:** Manages storage efficiently, reducing fragmentation and making full use of space.

Disadvantages of Using a File System

1. **Limited File Names:** Some systems restrict file name length or characters.
2. **Fragmentation:** Files can be scattered in non-contiguous blocks, slowing performance.
3. **Security Risks:** File systems can still be vulnerable if not properly configured.
4. **Wasted Space for Small Files:** Minimum block sizes can waste storage for tiny files.
5. **Complex Maintenance:** Large systems need regular tasks like defragmentation, cleanup, and backups.

File Access Methods:

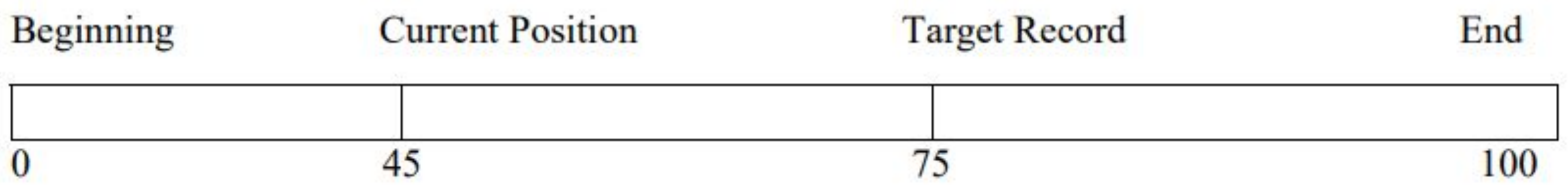
Files store information, and when needed, this information must be read into memory. There are different ways to access file data:

1. Sequential
2. Direct
3. Indexed

1. Sequential Access

1. The simplest access method.
2. Data is read in order, one record after another.
3. You cannot skip directly to a record; you must go through all previous records.
4. Sequential access is useful for storage like magnetic tapes.
5. Files can be rewound to read again from the beginning.

Example:
If a file has 100 records and the read/write head is at record 45, to read record 75, it must go through records 46, 47, … up to 75.



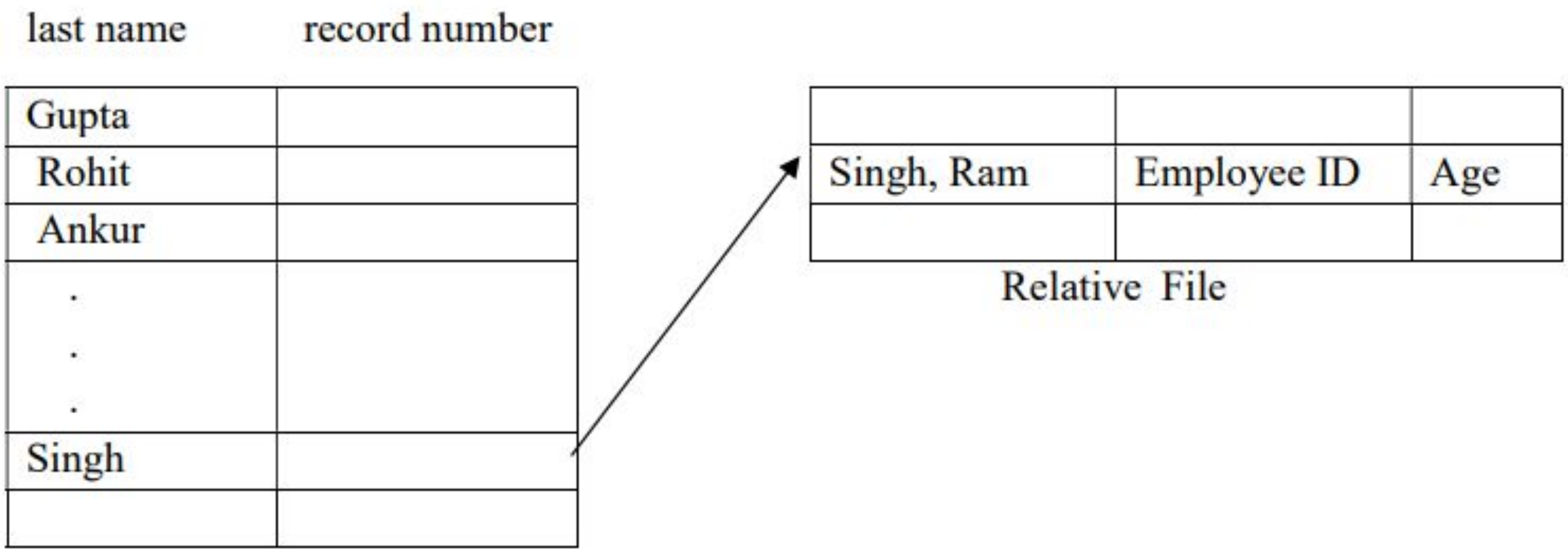
2. Direct (Random) Access

1. Files are made of fixed-length records, allowing quick reading and writing in any order.
2. Ideal for disk storage rather than sequential media like tapes.
3. Commonly used in applications like databases.
4. Example: An airline reservation system can access the record for a specific flight directly, without reading all earlier records.
5. You can read or write blocks in any order, e.g., read block 14, then block 50, then write block 7.
6. Very useful for fast access to large amounts of data.

3. Indexed Access

1. An index is created containing key fields and pointers to the actual data blocks.
2. To find a record, the system first searches the index and then uses the pointer to access the file directly.
3. Useful for large files, as it avoids scanning the entire file sequentially.
4. If the index itself becomes too large, a multi-level index can be used:
5. Primary index points to secondary indexes, which then point to the actual data.

Example: Systems like VMS (Virtual Memory Storage) use this method for efficient data access.


last name
record number
Gupta
Rohit
Ankur
.
.
.
Singh
Singh, Ram
Employee ID
Age
Relative File

Comparison of File Organizations

Feature	Sequential	Direct (Random)	Indexed
Access Method	Records accessed in order, one by one	Records accessed in any order	Records accessed via an index pointing to data
Speed	Slow for large files if specific record is needed	Fast, can directly access required record	Faster than sequential, slightly slower than direct for small files
Storage Medium	Suitable for tapes and sequential storage	Best for disks	Best for large disk-based files

Comparison of File Organizations

Feature	Sequential	Direct (Random)	Indexed
Ease of Use	Simple to implement	Moderate complexity	More complex due to index maintenance
Overhead	Low	Moderate	Higher due to index structures
Flexibility	Low	High	High, supports efficient searches
Ideal for	Processing all records in order	Immediate access to specific records	Searching and retrieving specific records efficiently

Use Cases and Efficiency

Sequential Access:

- **Use Cases:** Log files, batch processing, reading large datasets in order.
- **Efficiency:** Efficient for reading or writing all records in sequence, but slow for accessing a specific record.

Direct (Random) Access:

- **Use Cases:** Database records, airline reservation systems, inventory management.
- **Efficiency:** Very fast for accessing individual records, ideal for large files where records are frequently updated or retrieved in no particular order.

Indexed Access:

- **Use Cases:** Large database files, library catalogs, payroll systems.
- **Efficiency:** Efficient for searching and retrieving specific records, especially in very large files; slightly slower than direct access due to index lookup, but better than sequential scanning.

DISCUSSION & REVISION

1. Which file access method reads records one by one in order?
2. Which file access method allows reading records in any order?
3. In indexed access, what points to the actual data blocks?
4. Which file organization is best for tape storage?
5. Which access method is commonly used in databases?

REFERENCES

1. https://www.vbspu.ac.in/e-content/File-System-Management.pdf
2. https://www.scaler.com/topics/file-systems-in-os/