File Allocation Strategies in Operating Systems

1 Introduction

File allocation methods determine how files are stored on disk. Three primary strategies are:

- **Sequential Allocation**
- **Indexed Allocation**
- **Linked Allocation**

Each method has different ways of storing and managing disk space.

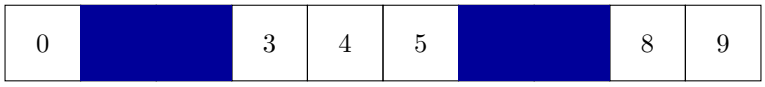
2 Sequential Allocation

In this method, a file occupies contiguous disk blocks. It is simple and fast but suffers from fragmentation.

2.1 Example Problem 1

Problem Statement: Given a disk with 10 blocks (0-9), some blocks are already allocated: {1, 2, 6, 7}. A file of size 4 needs allocation. Can it be allocated? If not, what happens?

Solution: No continuous space of 4 blocks is available. This results in **external fragmentation**. Compaction or a different allocation method is needed.



2.2 Example Problem 2

Problem Statement: A file of size 3 must be allocated, but only scattered free blocks are available (0, 3, 5, 8, 9). What issues arise?

Solution: Since sequential allocation requires continuous space, this file **cannot** be stored despite enough total space. External fragmentation occurs.

3 Indexed Allocation

Each file has an index block storing pointers to its disk blocks.

3.1 Example Problem 1

Problem Statement: A file requires 4 blocks. The index block is stored at block 1, but only three pointers fit per index block. How is the file stored?

Solution: If more pointers are needed, an indirect index block is required.

3.2 Example Problem 2

Problem Statement: What happens if the index block itself gets corrupted?

Solution: If the index block is lost, all file pointers are lost, making file recovery difficult unless backups exist.

4 Linked Allocation

Each block contains a pointer to the next, avoiding fragmentation but slowing access.

4.1 Example Problem 1

Problem Statement: A file is stored at locations $2 \rightarrow 6 \rightarrow 4 \rightarrow 9$, but block 6 gets corrupted. What happens?

Solution: Since pointers are stored within blocks, losing one block can break the entire chain unless redundancy is present.

4.2 Example Problem 2

Problem Statement: What happens if linked allocation is used for a large file and the file system does not allow backward traversal?

Solution: Backward traversal is inefficient, making random access slow.

5 Conclusion

Each allocation method has its advantages and drawbacks:

- **Sequential** is simple but has fragmentation issues.

- **Indexed** provides random access but requires an index block.
- **Linked** avoids fragmentation but is slow for random access.