1. Introduction

• Context

File management systems constitute a fundamental component of contemporary operating systems, acting as the essential interface between users and secondary storage devices. In the current digital environment, the effective organization and manipulation of data via file systems are integral to computing operations. This project is centered on the development of a simplified File Management System (FMS) simulator, which illustrates the primary mechanisms of file organization, storage allocation, and data manipulation in secondary memory.

• Objectives

The primary objectives of this project encompass both theoretical understanding and practical implementation:

- Master fundamental concepts of file management systems through hands-on implementation
- Develop proficiency in data structure manipulation within the context of file systems
- Implement and understand different memory allocation policies
- Create efficient algorithms for basic file operations (creation, insertion, search, deletion)
- Enhance C programming skills through practical application
- Gain experience in technical documentation and reporting

• Scope of the Project

The simulator implements a virtual secondary memory system with the following capabilities:

- Management of a block-based storage system with configurable size and blocking factor
- Support for both contiguous and chained file organization methods
- Implementation of metadata management for file tracking
- Basic file operations including creation, insertion, search, and deletion
- Memory management features such as defragmentation and compaction
- Interactive user interface through a command-line menu system

While the simulator provides a comprehensive overview of file system operations, it operates within certain constraints:

- Fixed-size record structure
- Limited to two file organization methods
- Simplified metadata management
- Basic memory allocation strategies

• Report Structure

This technical report is organized into the following sections:

1. Introduction: Provides context and project overview

2. System Design: Details the architectural choices and data structures

3. Implementation Details: Describes the algorithms and coding approaches

4. Testing and Results: Presents system validation and performance analysis

5. Conclusion: Summarizes achievements and potential improvements

Each section is designed to provide a comprehensive understanding of both the theoretical foundations and practical implementation of our File Management System simulator.

2. System Design and Architecture

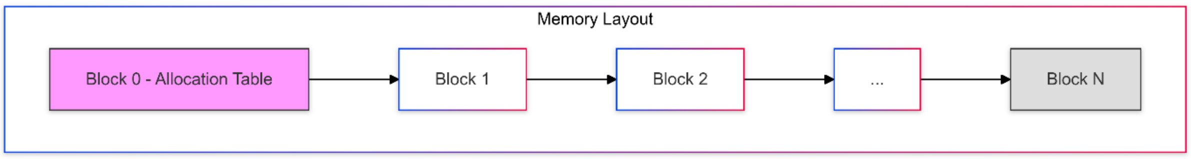
2.1 Memory Model and Organization

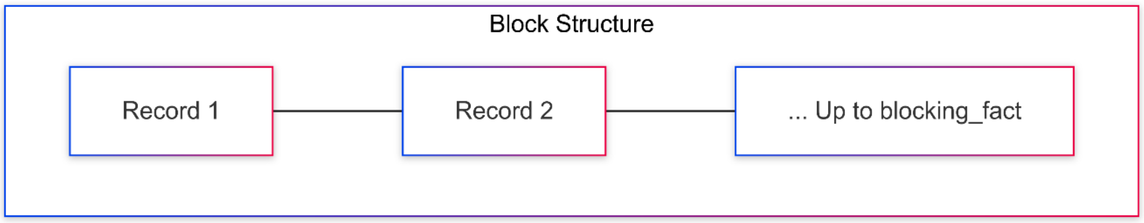
Our file management system implements a flexible block-based architecture that allows users to define both the total number of blocks and blocking factor during system initialization. This design choice provides significant advantages for different use cases, as organizations can tailor the storage system to their specific needs. The system reserves the first block for the allocation table, which maintains the status of all subsequent blocks and serves as the central management unit for storage allocation.

• Secondary Storage Structure

The **metadata file** serves as the central management unit, storing essential information such as the number of blocks, block capacity, and file organization details. Each data file references the metadata to determine block length, addresses, and allocation status, ensuring streamlined storage control.

The **virtual disk blocks** function as fixed-size storage units, where each block can contain a user-defined number of records. Each block header stores the necessary pointers and allocation data, while the data section holds the actual file records. This design supports efficient space management and offers flexibility for handling varying file sizes.





• Organization Methods

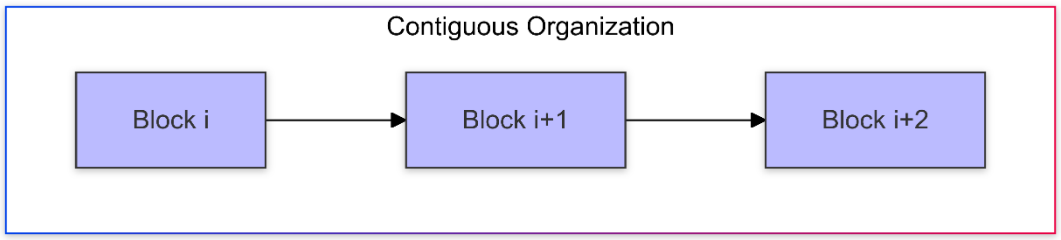
The system supports two block organization methods:

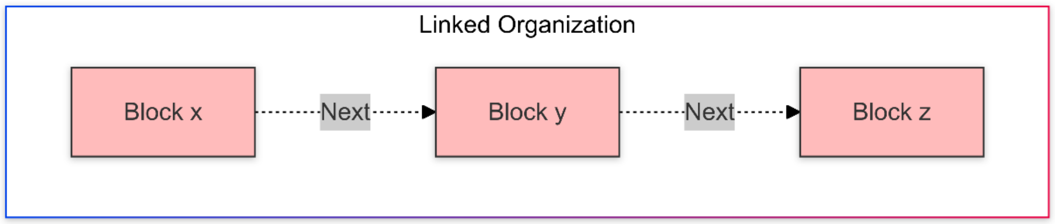
- **Contiguous Organization:** Files occupy consecutive blocks, providing optimal performance for sequential data access and minimal fragmentation.
- **Linked Organization:** Files are stored in non-consecutive blocks, each linked via pointers, maximizing space efficiency and flexibility for dynamic file sizes.

Additionally, the system provides two **internal file organization** modes, defined in the metadata file using a binary flag (0 or 1):

- **Sorted (1):** Records within a block are stored in a specific order based on an identifier, facilitating faster search operations.
- **Unsorted (0):** Records are stored as they arrive, prioritizing simplicity and faster insertions over optimized search performance.

This dual-file architecture, combined with both block and internal file organization flexibility, ensures efficient storage control and adaptability for diverse data management scenarios.





2.2 Data Structures and Justification

The implementation relies on three core data structures, each carefully designed to balance efficiency with simplicity. Our choice of data structures reflects the need for both performance and maintainability, while ensuring the system remains accessible for educational purposes.

- ### Record Management

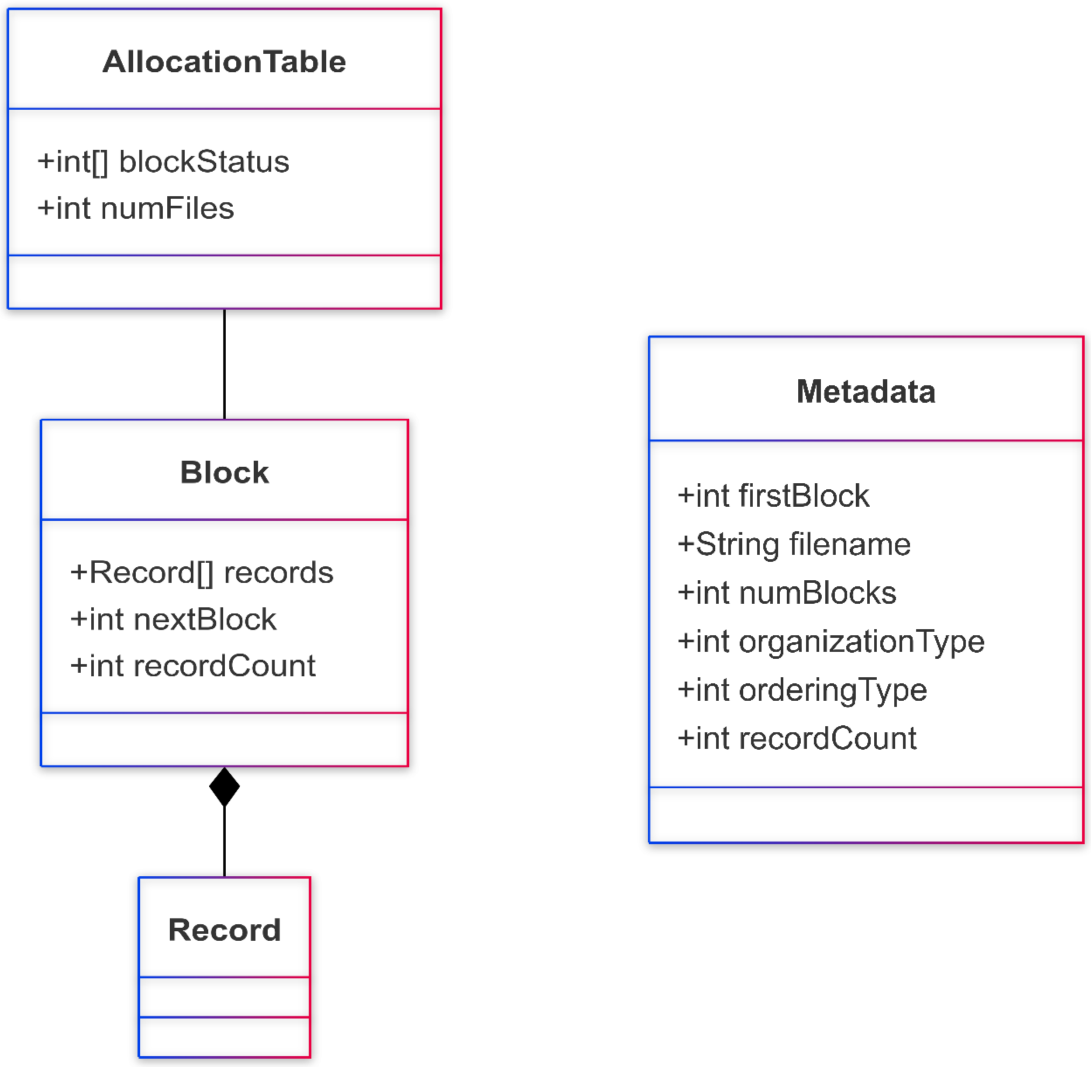
Records serve as the basic data unit, implementing:

- A unique identifier for sorting and retrieval
- A deletion flag for efficient space management
- Fixed size within each block based on the blocking factor

- ### Block Organization

Blocks function as the primary storage units with:

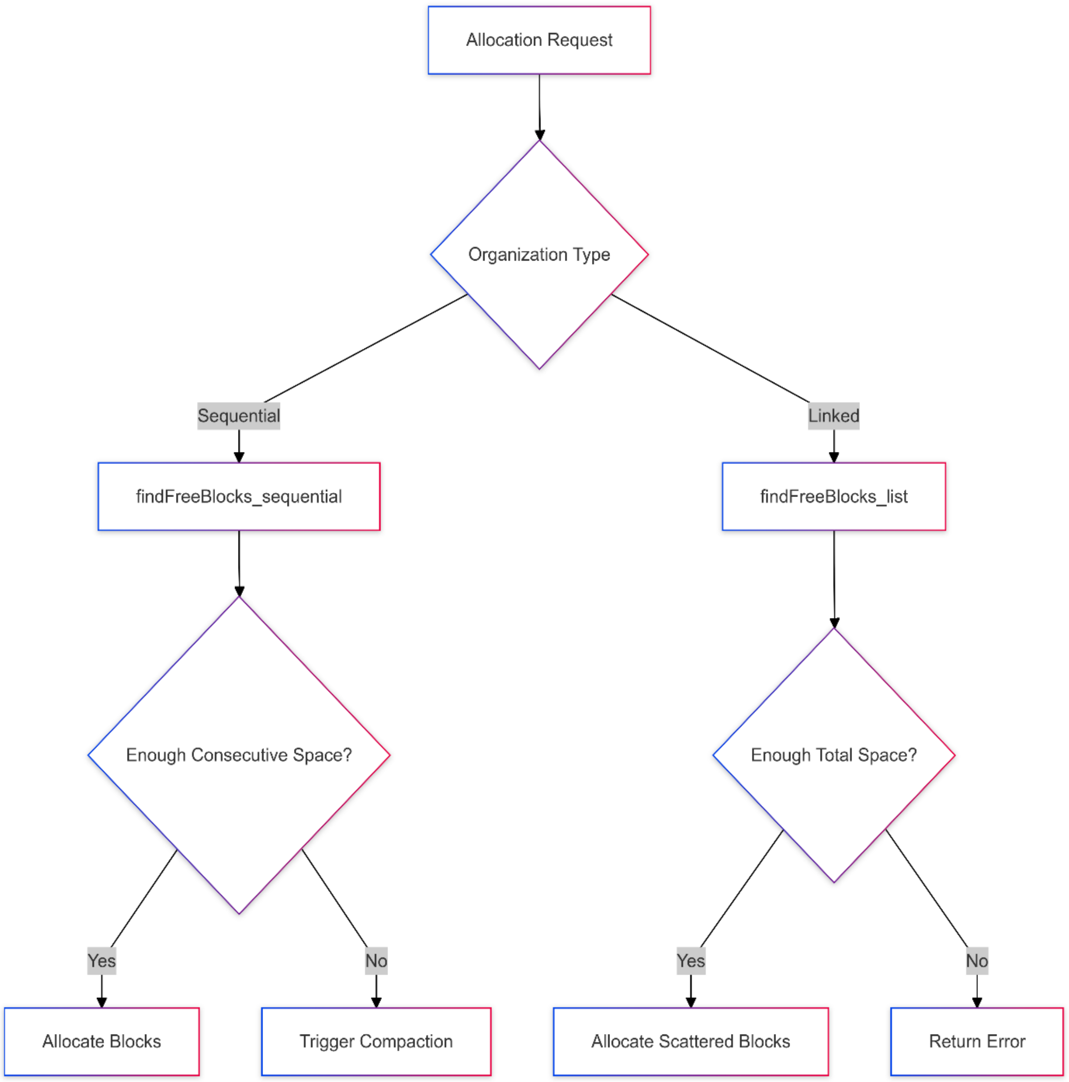
- Dynamic record array sized according to the blocking factor
- Next block pointer enabling linked organization
- Record counter for partial block utilization



2.3 Algorithms and Functionalities

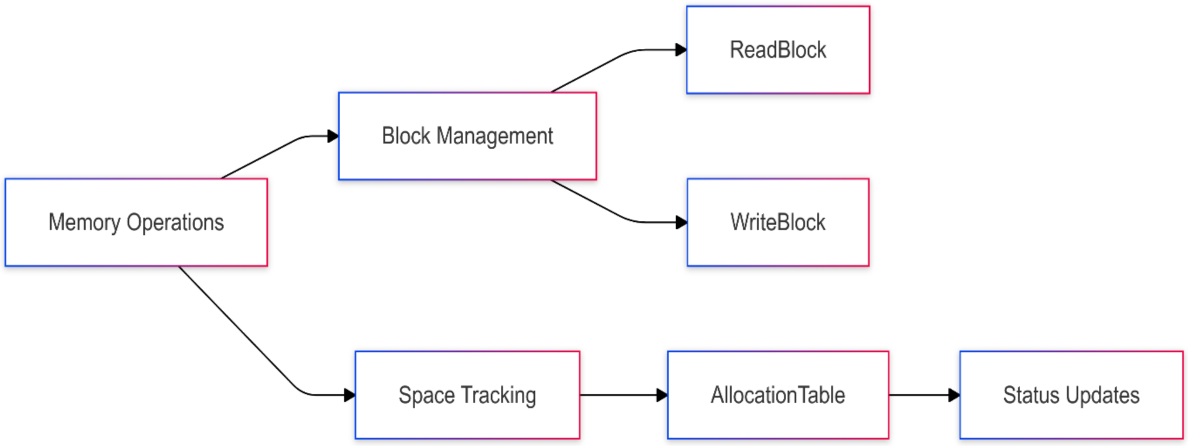
- ### Block Allocation Strategy

The system uses algorithms for dynamic block allocation and file organization, balancing efficiency and data integrity. It employs sequential and linked allocation methods. The sequential approach uses a first-fit strategy for speed, though it can cause fragmentation, while linked allocation allows non-contiguous block use for better space efficiency.



- ## Memory Management Workflow

Dynamic memory management maximizes space use and data integrity. The metadata system records block allocation, organization, and record details, ensuring consistency and fast data retrieval.



- ## Implementation Details

The system implements several key operations that work with the dynamic block sizing:

- Initialization

- AllocationTable initialization adapts to the specified block count
- Block structures are created according to the defined blocking factor
- Memory allocation validates available system resources

- Block Operations

- Read and write operations calculate offsets based on dynamic block sizes
- Block access functions handle variable-sized records efficiently
- Display functions adapt to different blocking factors

- Memory Management

- Allocation tracking scales with total block count
- Free space search algorithms work with variable block counts
- Compaction procedures handle different block sizes

This architecture ensures reliable file management while maintaining flexibility through configurable block sizes. The system's modular design allows for easy adaptation to different storage requirements while maintaining operational efficiency.