Indexed File System Simulator

Name: Arnav Meshram

Roll No: 2303101

Course: CS310 Project

Institution: Indian Institute of Technology Goa

What is a File Management System?

A **File Management System** is a software component that manages how data is stored, organized, and retrieved on storage devices.

📁 Data Organization

Structures data in a logical, hierarchical manner

⚡ Fast Access

Quickly locates and retrieves data

🔒 Safety

Protects data through access controls

💾 Efficient Allocation

Optimizes disk space usage

File Allocation Methods

Contiguous

Files occupy consecutive blocks

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

Fast but fragmentation issues

Linked
Blocks point to next block

No fragmentation, slow access

Indexed
Index block stores all addresses

Fast access, no fragmentation

Indexed File Allocation

Uses an **index block** that stores addresses of all data blocks belonging to a file.

How it works:

- Each file has its own index block
- Index contains addresses of all data blocks
- Enables direct access to any block
- No need to traverse links

Visual:

Index → **10** **15** **22**

Project Overview

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

This project simulates an Indexed File Allocation system using Python and Tkinter.

System Config

Total Blocks: 64
Block Size: 4096 bytes
Capacity: 256 KB

Core Features

Create/Delete files
Write/Read content
View inode table
Track blocks

```
Data Structures:
free_blocks[] - Boolean array (64 elements)
files{} - Dictionary storing file metadata
```

Key Functions

allocate_blocks()

Allocates 1 index block + required data blocks.

```
1. Calculate blocks needed (data + 1 index)
2. Check if enough free blocks exist
3. Allocate first free as index block
4. Allocate rest as data blocks
5. Mark blocks as allocated in free_blocks[]
```

create_file(name, size_kb)

- Checks if file already exists
- Calculates required blocks (ceiling division)
- Calls allocate_blocks()
- Stores file info in files dictionary

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

delete_file(name)

- Verifies file exists
- Frees index block
- Frees all data blocks
- Removes file from dictionary

write_file() & read_file()

- **write_file:** Saves content to file's content field
- **read_file:** Retrieves and returns file content

get_inode_table()

Returns file information: name, size, index block, and data blocks.

Tkinter GUI

Control Buttons:

- **Create File** - Input name and size in KB
- **Delete File** - Remove file and free blocks
- **Write File** - Save content to file
- **Read File** - Display file content
- **Show Inode Table** - Display all file metadata
- **Show Free Blocks** - Display block status

Free Block Display

Shows all 64 blocks in groups of 8:

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

```
Blocks 00-07: F F A A F F F F
F = Free | A = Allocated
```

F F A A F

Example Workflow

1. Create file "doc.txt" (10 KB)
→ System calculates: 10KB ÷ 4KB = 3 blocks needed
→ Allocates: 1 index + 3 data = 4 total blocks
→ Result: Index=0, Data=[1,2,3]

2. Write content "Hello World"
→ Stores in file's content field

3. View Inode Table
→ Shows: doc.txt | 10 KB | Index: 0 | Blocks: [1,2,3]

4. Delete file
→ Frees blocks 0, 1, 2, 3
→ All blocks available again

Advantages

⚡ Fast Access

🎯 No Fragmentation

Blocks can be anywhere on disk

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

Direct access to any block via index

📈 Dynamic Growth

Files can expand easily

👁️ Visual Feedback

Real-time block status display

Conclusion

This project successfully demonstrates:

- Implementation of indexed file allocation
- Efficient block management system
- User-friendly GUI interface
- Core operating system concepts

Thank You!

Hello World!

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF