University Of Science And Technology Houari Boumediene

2nd year computer engineering

File management system -report

Authors:[Louail Wissam & Moussaoui Meriem Nada]

Section B

Objective

The objective of this code is to implement a simplified file management system that supports fundamental operations like file creation, insertion, searching, deletion (logical and physical), defragmentation, and metadata management. It also incorporates a user authentication mechanism with login and sign-up functionality.

Code Overview

1. Menu Function

The menu() function displays a list of available operations for the user. It includes functionalities like initializing secondary memory, creating and managing files, searching and managing records, defragmenting files, compacting memory, and exiting the program. The menu offers 14 options for comprehensive file system management.

2. Main Functionality

The main() function drives the program. Key operations include:

- **Authentication**: The user must log in or sign up to access the system.
- **File Management Operations**: A switch-case structure executes the selected operation based on user input.

3. Key Components

- **Authentication**:
  - login() and signup() functions manage user authentication. This ensures secure access to the file management system.
- **File Handling**:
  - Two key files are used:
    - MS.bin: Represents the secondary memory.
    - MetaDonnees.data: Stores metadata about files and records.
  - Both files are created and opened in binary mode for efficient storage and retrieval.
- **Operations on Files**:
  - The program supports a wide range of file management tasks like creating, loading, renaming, and deleting files.
- **Memory Management**:
  - Functions like instializems() and compactdisk() handle secondary memory initialization and compaction.

Functionalities

Authentication

•**Log In (Option 1)**: Verifies user credentials.
•**Sign Up (Option 2)**: Registers a new user.

File Operations

1.Initialize Secondary Memory: Prepares the storage for use.
2.Create and Load File: Creates a new file and loads it into memory.
3.Display SM Status: Graphically displays the current status of secondary memory.
4.Show File Metadata: Displays metadata associated with files.
5.Search Record in File: Searches for a specific record in a file based on a key.
6.Insert Record: Adds a new record to a file.
7.Logical Deletion: Marks a record as deleted without physically removing it.
8.Physical Deletion: Permanently removes a record from a file.
9.Defragmentation: Reorganizes blocks within a file to optimize storage.
10.Delete File: Removes a file and its metadata.
11.Rename File: Changes the name of a file.
12.Compact Memory: Optimizes storage by consolidating data.
13.Clear Secondary Memory: Deletes all files and resets memory.
14.Exit Program: Terminates the application.

Code testing

Test Case 1: Create a New File

Objective:

• Test the functionality to create a new file named TestFile and ensure it is added to the metadata.

Expected Results:

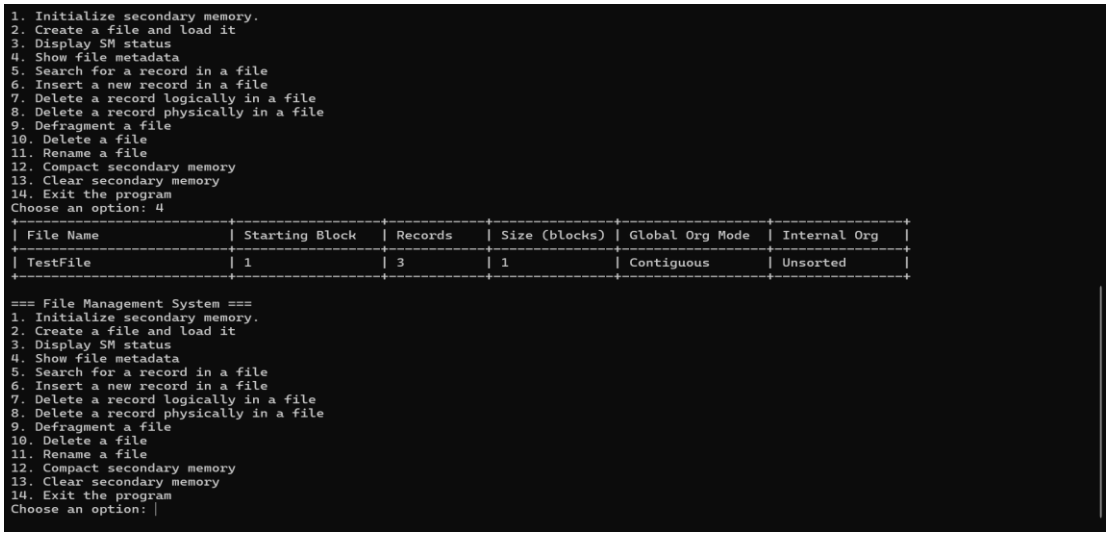
•A file named TestFile is created and added to the metadata.

•The file should appear in the metadata when listing files using option **4. Show file metadata**.

Actual Results:

•The file TestFile was created successfully.

• Upon selecting option 4. Show file metadata, the metadata for TestFile was displayed



Test Case 2: Insert a Record into a File

Objective:

• Test the functionality to insert a record with key 101 and value SampleValue into the file TestFile.

Expected Results:

• The record with key 101 and value SampleValue is successfully inserted into TestFile.

• The file's metadata reflects the new record.

Actual Results:

• The record was successfully inserted into TestFile.

```
=== File Management System ===
1. Initialize secondary memory.
2. Create a file and load it
3. Display SM status
4. Show file metadata
5. Search for a record in a file
6. Insert a new record in a file
7. Delete a record logically in a file
8. Delete a record physically in a file
9. Defragment a file
10. Delete a file
11. Rename a file
12. Compact secondary memory
13. Clear secondary memory
14. Exit the program
Choose an option: 6
Enter key: 101
Enter value: SampleValue
Enter the file name to insert the record: TestFile
Enter file name to load: TestFile
Metadata: Address = 1, Size = 1, GlobalOrg = 0, InternalOrg = 0
Record not found. Should be at Block: 1, Offset: 3.
Record with ID 101 successfully inserted into file 'TestFile'.
Record inserted successfully.

=== File Management System ===
1. Initialize secondary memory.
2. Create a file and load it
3. Display SM status
4. Show file metadata
5. Search for a record in a file
6. Insert a new record in a file
7. Delete a record logically in a file
8. Delete a record physically in a file
9. Defragment a file
10. Delete a file
11. Rename a file
```

• Metadata inspection via option **4. Show file metadata** displayed the updated file details.

```
1. Initialize secondary memory.
2. Create a file and load it
3. Display SM status
4. Show file metadata
5. Search for a record in a file
6. Insert a new record in a file
7. Delete a record logically in a file
8. Delete a record physically in a file
9. Defragment a file
10. Delete a file
11. Rename a file
12. Compact secondary memory
13. Clear secondary memory
14. Exit the program
Choose an option: 4
+-------------------------+-----------------+------------+---------------+-----------------+----------------+
| File Name               | Starting Block  | Records    | Size (blocks) | Global Org Mode | Internal Org   |
+-------------------------+-----------------+------------+---------------+-----------------+----------------+
| TestFile                | 1               | 4          | 2             | Contiguous      | Unsorted       |
+-------------------------+-----------------+------------+---------------+-----------------+----------------+

=== File Management System ===
1. Initialize secondary memory.
2. Create a file and load it
3. Display SM status
4. Show file metadata
5. Search for a record in a file
6. Insert a new record in a file
7. Delete a record logically in a file
8. Delete a record physically in a file
9. Defragment a file
10. Delete a file
11. Rename a file
12. Compact secondary memory
13. Clear secondary memory
14. Exit the program
Choose an option:
```

Test Case 3: Logical Deletion of a Record

Objective:

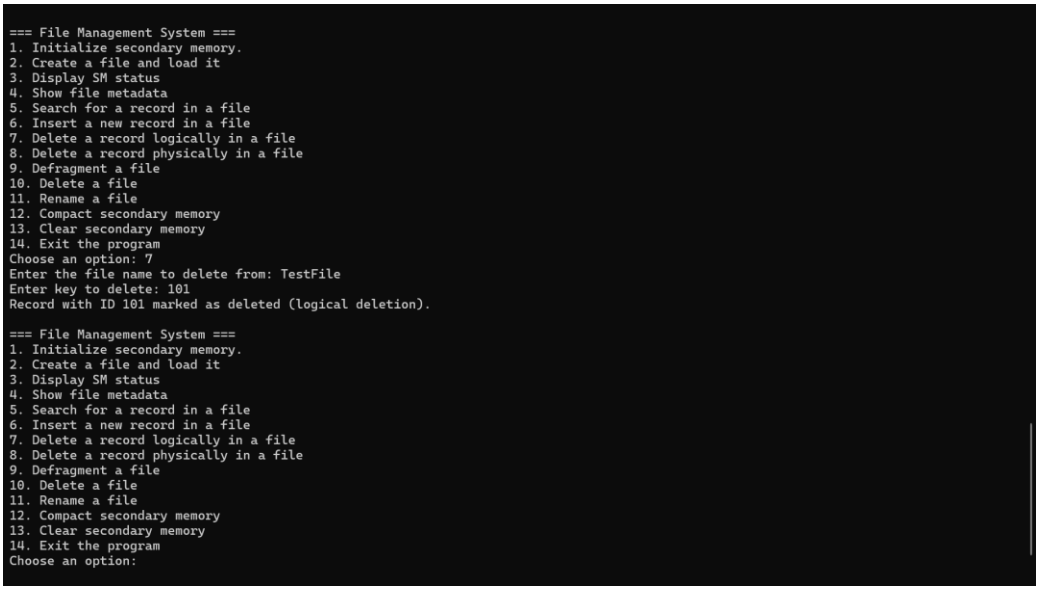
• Test the functionality to logically delete the record with key 101 from TestFile.

Expected Results:

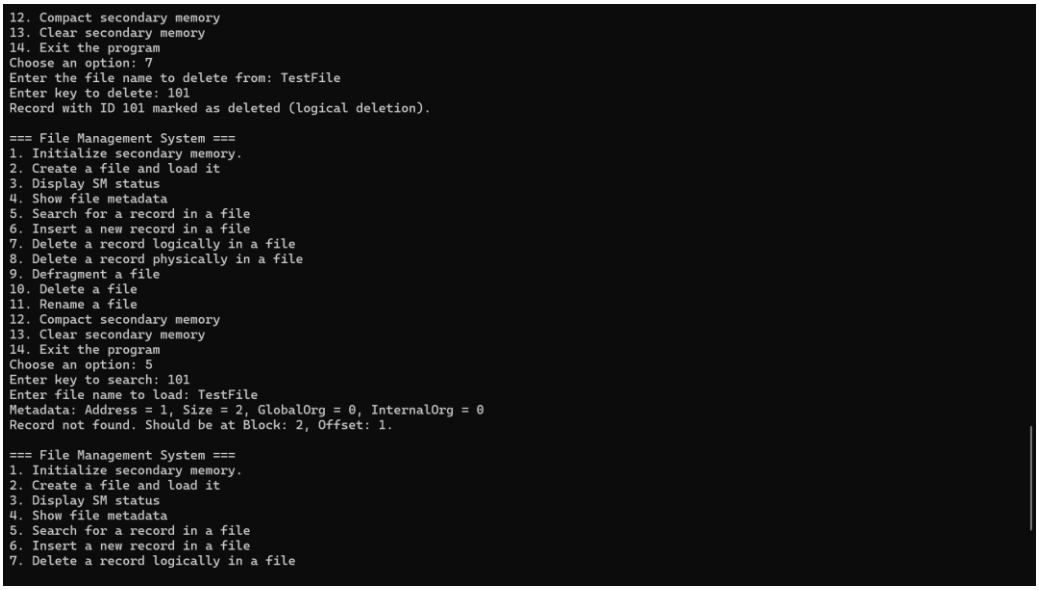
• The record with key 101 is marked as deleted but remains in the file for potential restoration.

Actual Results:

• The record was logically deleted successfully.



• Subsequent searches for key 101 confirm its logical deletion.



Test Case 4: Display the memory status

Objective:

• Test the functionality Display ms to check if it works.

Expected Results:

• Before creating the file TestFile all blocks in ms are free.

• After creating the file TestFile block number 2 and block number 3 are occupied and has the file in them (since its size is 2 block).

Actual Results:

• all blocks in ms are free before creating any file.

• block number 2 and block number 3 are occupied and has the file in them

Strengths

• **Comprehensive Features:** Covers a wide range of file and memory management functionalities.
• **User Authentication:** Adds a layer of security to the system.
• **Structured and Modular Design:** Logical separation of functionalities enhances readability and maintainability.
• **Dynamic Interaction:** Uses user input to dynamically execute desired operations.

Conclusion

This project successfully modeled a file management system by simulating key mechanisms such as block allocation, space management, and file manipulation. The objectives were achieved, although improvements are possible, particularly in performance for large datasets and dynamic modifications .

Team Members

MOUSSAOUI Meriem Nada | 232331432803 | Groupe 4
LOUAIL Wissam | 232331500218 | Groupe 4