Rapport

Introduction

The task involves implementing several operations for managing file records in a simplified file system simulator. These operations include:

- Searching for a record by ID.
- Logical deletion of a record.
- Physical deletion of a record.
- File defragmentation to remove logically deleted records.
- Renaming a file while updating associated metadata.

This report explains the implementation, pseudocode, and examples for each operation.

Functionality Overview

1. Search for a Record by ID

This function searches for a specific record in a file by its unique ID and displays the record's details if found.

Pseudocode

```
Open the file in read mode.
For each record in the file:
    If the record ID matches the target ID:
        Display record details.
        Exit.
If no match is found:
    Display "Record not found".
Close the file.
```

Example

Input: File `data.dat` containing:

```
ID: 1, Name: Alice, Address: Paris
ID: 2, Name: Bob, Address: Berlin
```

Search for: ID = 2.
Output: "Record found: ID: 2, Name: Bob, Address: Berlin".

2. Logical Deletion of a Record

This function marks a record as deleted by setting its ID to -1, while keeping the record in the file.

Pseudocode

```
Open the file in read-write mode.
For each record in the file:
    If the record ID matches the target ID:
        Set record ID to -1.
        Write the updated record back to the file.
        Exit.
If no match is found:
    Display "Record not found".
Close the file.
```

Example

Input: File `data.dat` containing:

```
ID: 1, Name: Alice, Address: Paris
ID: 2, Name: Bob, Address: Berlin
```

Delete logically: ID = 1.
Output: File `data.dat` updated to:

```
ID: -1, Name: Alice, Address: Paris
ID: 2, Name: Bob, Address: Berlin
```

3. Physical Deletion of a Record

This function removes a record from the file permanently by rewriting the file without the specified record.

Pseudocode

```
Open the file in read mode.
Open a temporary file in write mode.
For each record in the file:
    If the record ID does not match the target ID:
```

```
        Write the record to the temporary file.
Replace the original file with the temporary file.
Close both files.
```

Example

Input: File `data.dat` containing:

```
ID: 1, Name: Alice, Address: Paris
ID: 2, Name: Bob, Address: Berlin
```

Delete physically: ID = 1.
Output: File `data.dat` updated to:

```
ID: 2, Name: Bob, Address: Berlin
```

4. File Defragmentation

This function removes all logically deleted records (marked with ID = -1) and compacts the file.

Pseudocode

```
Open the file in read mode.
Open a temporary file in write mode.
For each record in the file:
    If the record ID is not -1:
        Write the record to the temporary file.
Replace the original file with the temporary file.
Close both files.
```

Example

Input: File `data.dat` containing:

```
ID: -1, Name: Alice, Address: Paris
ID: 2, Name: Bob, Address: Berlin
```

Defragment:
Output: File `data.dat` updated to:

```
ID: 2, Name: Bob, Address: Berlin
```

5. File Renaming

This function renames a file and updates the metadata to reflect the new name.

Pseudocode

```
Rename the file using system call.
If successful:
    Update metadata to reflect the new file name.
    Display success message.
Else:
    Display error message.
```

Example

Input: File name: `data.dat`.
Rename to: `records.dat`.
Output: "File 'data.dat' successfully renamed to 'records.dat'."

Conclusion

The implementation of these file operations ensures robust functionality with error handling and user feedback. These tasks are critical for managing records in the simplified file system simulator.