FILE SYSTEM DESIGN – MACHINE CODING PROBLEM STATEMENT

Goal:

Design and implement an in-memory file system that supports creation, linking, appending, deletion, and movement of files and directories.

Functional Requirements:

1. Add a File

• Input: Full file path.

• Automatically create intermediate directories (if not present).

• Create a new file at the deepest level.

Example:

addFile("/a/b/c/file1")

2. Append to a File

• Append content to an existing file.

• If the file does not exist → return error.

Example:

append("/a/b/c/file1", "hello")

3. Create a Linked File (Hard Link)

• Create a new file reference pointing to the same physical file.

• Both files share the same underlying content.

Example:

link("/a/b/c/file1", "/x/y/newFile")

4. Append to a Linked File

• Appending to any link updates the same shared content.

Example:

append("/x/y/newFile", "foo")

5. Delete Link

• Delete one reference to a file.

• If this is the last reference, delete the file data.

Example:

delete("/x/y/newFile")

6. Move File

• Move a file reference from one path to another.

• If the file has multiple links, only the reference moves.

Example:

move("/a/b/c/file1", "/p/q/fileZ")

Clarifications:

• Maintain hierarchical directory structure.

• A file may have multiple hard links; all share content.

• linkCount tracks active file references.

• When linkCount becomes 0, file data is deleted.

• No symbolic/soft links required.

• ASCII file content only.

• Directory deletion not required.

• Thread safety not required.