Names: M Ahsan Rehman __Shaheer Ayyaz__Taqdees Mir

Roll No: FA22-BSE-033 _FA22-BSE-015_FA22-BSE-041

Submitted to: Sir Sami-Ullah

Mirpur University of Science & Technology
Department of Software Engineering

Critical Analysis of Formal Correctness and Verification of Simple File Systems Using Z Schema

Formal methods, such as the Z notation, are used to mathematically model and verify the correctness of software systems. In this analysis, we will use Z schema to specify a simple file system, define its operations, and verify their correctness. The Z notation provides a precise way to describe the state of a system and its operations, ensuring that the system behaves as intended.

1. Z Schema for a Simple File System

Basic Definitions

A file system can be modeled as a collection of files, where each file has a unique identifier and contains data. We define the following components:

- **File**: A file is represented as a sequence of data.
- **FileSystem**: A collection of files indexed by unique identifiers.

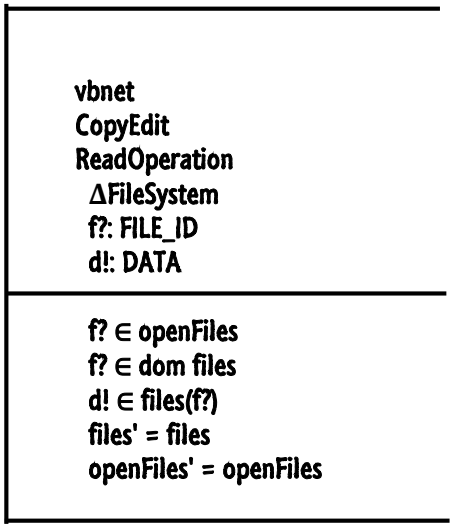
Z Schema: File System State

2. Operations on the File System

2.1 Read Operation

The read operation retrieves data from a file. The file must be open and exist in the file system.

Z Schema: Read Operation



Preconditions:

- The file f? must be open (f? ∈ openFiles).
- The file f? must exist in the file system (f? ∈ dom files).

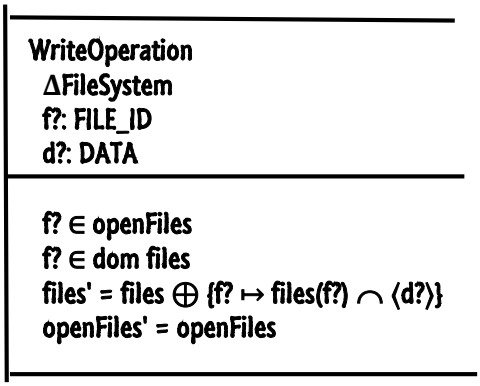
Postconditions:

- The data d! is retrieved from the file f? (d! ∈ files(f?)).
- The state of the file system remains unchanged (files' = files, openFiles' = openFiles).

2.2 Write Operation

The write operation appends data to a file. The file must be open and exist in the file system.

Z Schema: Write Operation



Preconditions:

- The file f? must be open (f? ∈ openFiles).
- The file f? must exist in the file system (f? ∈ dom files).

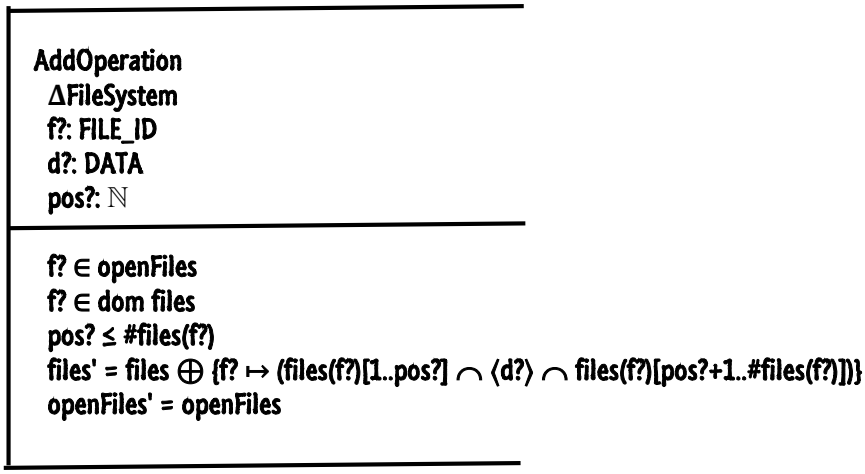
Postconditions:

- The data d? is appended to the file f? (files' = files ⊕ {f? ↦ files(f?) ⌢ ⟨d?⟩}).
- The state of the file system is updated to include the new data.

2.3 Add Operation

The add operation inserts data at a specific position in a file. The file must be open and exist in the file system.

Z Schema: Add Operation



Preconditions:

- The file f? must be open (f? ∈ openFiles).
- The file f? must exist in the file system (f? ∈ dom files).
- The position pos? must be within the bounds of the file (pos? ≤ #files(f?)).

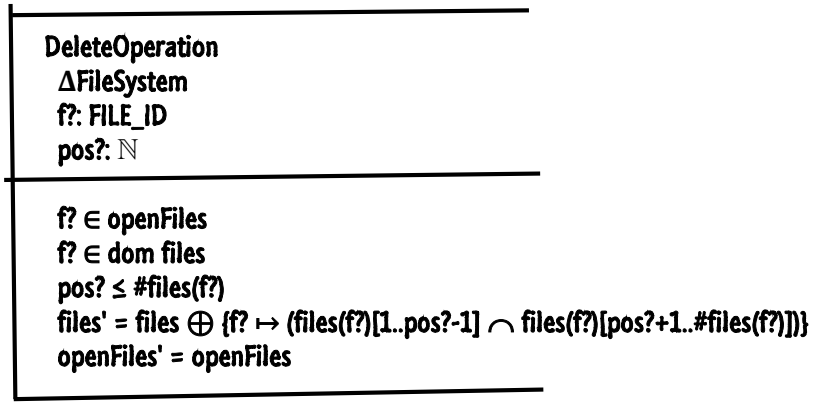
Postconditions:

- The data d? is inserted at position pos? in the file f?.

2.4 Delete Operation

The delete operation removes data from a file. The file must be open and exist in the file system.

Z Schema: Delete Operation



Preconditions:

- The file f? must be open (f? ∈ openFiles).
- The file f? must exist in the file system (f? ∈ dom files).
- The position pos? must be within the bounds of the file (pos? ≤ #files(f?)).

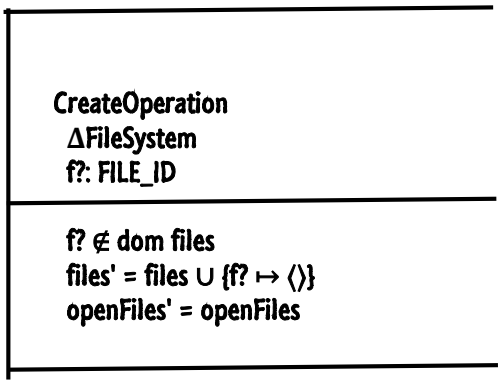
Postconditions:

- The data at position pos? is removed from the file f?.

2.5 Create Operation

The create operation adds a new file to the file system.

Z Schema: Create Operation



Preconditions:

- The file f? must not already exist in the file system (f? ∉ dom files).

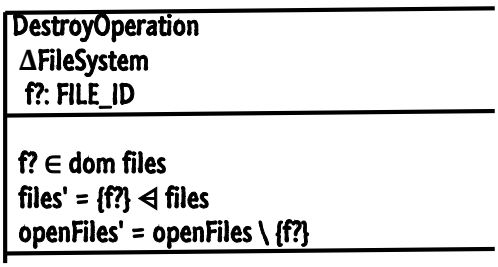
Postconditions:

- A new file f? is added to the file system with empty content (files' = files ∪ {f? ↦ ⟨⟩}).

2.6 Destroy Operation

The destroy operation removes a file from the file system.

Z Schema: Destroy Operation



Preconditions:

- The file f? must exist in the file system (f? ∈ dom files).

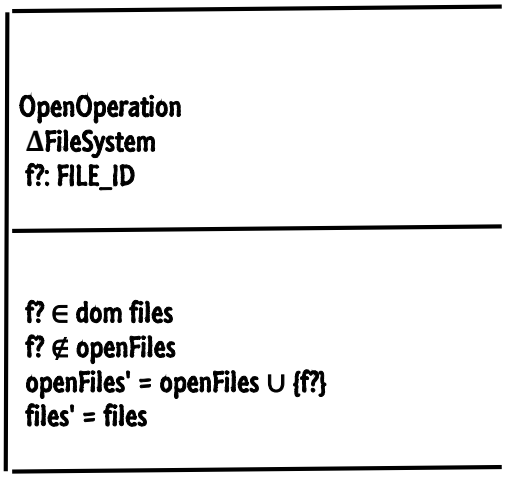
Postconditions:

- The file f? is removed from the file system (files' = {f?} ⩤ files).
- If the file was open, it is removed from the set of open files (openFiles' = openFiles \ {f?}).

2.7 Open Operation

The open operation makes a file available for reading and writing.

Z Schema: Open Operation



Preconditions:

- The file f? must exist in the file system (f? ∈ dom files).
- The file f? must not already be open (f? ∉ openFiles).

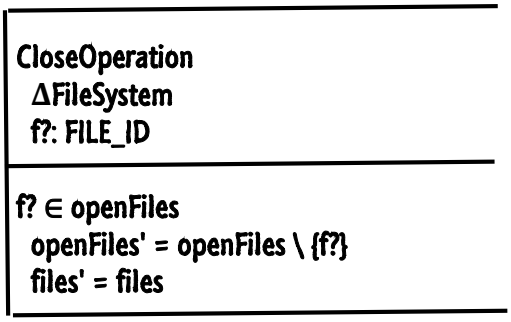
Postconditions:

- The file f? is added to the set of open files (openFiles' = openFiles ∪ {f?}).

2.8 Close Operation

The close operation makes a file unavailable for reading and writing.

Z Schema: Close Operation



Preconditions:

- The file f? must be open (f? ∈ openFiles).

Postconditions:

- The file f? is removed from the set of open files (openFiles' = openFiles \ {f?}).

3. Promoting a Single File to an Indexed Component of a File System

To promote a single file to an indexed component of the file system, we use the files mapping in the FileSystem schema. Each file is uniquely identified by its FILE_ID, and the files function maps each FILE_ID to its corresponding data sequence.

4. Verification of Formal Correctness

The Z schemas provide a formal specification of the file system and its operations. Automated tools like Z/EVES or ProZ can be used to:

- Verify the consistency of the schemas.
- Check that the preconditions and postconditions are correctly defined.
- Ensure that the operations preserve the invariants of the file system (e.g., no duplicate file IDs, no invalid file access).

Conclusion

Using Z schema notation, we have formally specified a simple file system and its operations. This mathematical modeling ensures the correctness and reliability of the file system. Automated tools can further verify the formal correctness of the system, making it suitable for critical applications where precision is essential.