File system integrity in Linux

The goal of this assignment is to get familiar with Linux file system APIs. In this assignment, you are to implement a secure library (`SecureFS`) on top of the existing file system interfaces that raises the alarm if the integrity of the files created through the secure library APIs is compromised. At the high-level, `SecureFS` maintains a Merkle tree for every file to check the consistency of file blocks before every read and write. The root of the Merkle tree is saved on disk to verify the integrity of files after reboot.

1 Cryptographic hash function

A cryptographic hash function generates a 20 bytes hash value of a given string of any length. The hash is computed in such a manner that it is infeasible to find two strings whose hash value are the same.

2 Merkle tree

A Merkle tree is a tree of hashes of data blocks. The file is divided into data blocks (you have to use 64 bytes data blocks) of fixed size. The leaf nodes of a Merkle tree are the hash values of file blocks. An internal node of a Merkle tree is the hash of concatenation of hashes of its child nodes. The root of the Merkle tree is the unique hash of the entire file.

3 Integrity check

To check the integrity of file, `SecureFS` computes a unique hash value from the file contents and store in `secure.txt` file. `SecureFS` assumes that `secure.txt` cannot be tampered. When a file is opened, `SecureFS` creates a Merkle tree (in memory) from the file blocks. `secure.txt` contains the root of the Merkle tree corresponding to every file created by the `SecureFS` interface. Whenever a file is modified the Merkle tree is updated, and the root of the Merkle tree is synced with the `secure.txt`. The in-memory Merkle tree is deleted when the file is closed.

4 Implementation

Read the man pages of `open`, `close`, `read`, `write`, and `lseek` system calls. Download the base files using:
`git clone https://github.com/Systems-IIITD/filesys.git`

`get_sha1_hash` returns a 20 bytes hash value of an input buffer of a given length. The nodes of the Merkle tree contains the 20 bytes hash returned through `get_sha1_hash` API. To create a Merkle tree, you have to divide the files into 64 bytes blocks. You are to implement the following interfaces in `filesys.c`.

- `filsys_init`: `filesys_init` creates the `secure.txt` file if it doesn't exist. It also checks the integrity of all the files whose hashes are present in `secure.txt`. If a file doesn't exist, `filesys_init` removes the corresponding entry from `secure.txt`. If the integrity of an existing file is compromised `filesys_init` returns 1. `filesys_init` returns 0 on success.
- `s_open`: `s_open` builds the Merkle tree from the file data and compares the root hash with the one stored in `secure.txt`. `s_open` returns -1, if the integrity check fails. If the file doesn't exist, a new entry is created in `secure.txt`. If the file is going to be truncated, `s_open` updates the Merkle tree and `secure.txt` entry accordingly.
- `s_read`: `s_read` computes the blocks of the file that need to be read. After reading these blocks `s_read` checks the integrity using the Merkle tree. If the integrity check fails, then -1 is returned to the caller.
- `s_write`: Before writing, `s_write` checks the integrity of file blocks that are going to be modified. On failing the integrity check, -1 is returned to the caller. `s_write` updates the Merkle tree, synchronize root hash with `secure.txt` and write modified blocks of the file.
- `s_lseek`: `s_lseek` ensures that `SEEK_END` points to the size of the file updated through the `SecureFS` APIs.

5 Grading

Execute `‘‘make && make run’’` in the `filesys` folder to run the test cases. The filesys folder contains four test cases. Each test case carries one mark. The design documentation is of two marks. You are eligible for design documentation marks if all the test cases are passing.

6 Design documentation

Read the man page of `fsync` system call. Answer the following questions.

- Which test cases are failing? If none, mention all test cases are passing?
- Will you be able to check the integrity of all files after a crash (power failure). If yes, please discuss how your design ensures consistency after a crash. If no, suggest a scheme using standard file system APIs (including fsync), that ensures the consistency of files contents with the stored hashes after a power failure.

7 Submission

This is a group assignment. Upload a pdf file of your design documentation at the provided link. You must follow the naming convention as group_id.pdf. Upload the entire `filesys` folder at the submission link.