ToyFS - Implement proper locking

Filesystems can be accessed by several users/processes concurrently, creating several concurrent internal operations within the filesystem, like creation and deletion of inodes, allocating and freeing blocks, directory entries, etc.
The metadata data structures used by the filesystem should be properly protected, so that concurrent access does not corrupt the filesystem.
ToyFS relies mostly on generic implementations which provide inode synchronization already. However the superblock can still be updated concurrently. Hence, your task is to ensure all accesses/updates to the superblock are properly synchronized.

What to Submit

- Patch(es) implementing access synchronization to ToyFS superblock
- It is up to you to decide what should be protected and how.

Procedure

- Clone the ToyFS source code
- Ensure it builds against Linux kernel 6.16
- Implement access synchronization to superblock structure

Requirements

- Protect the filesystem against concurrent access to the superblock
- Patches should have proper comments to the code explaining the locking mechanism and the lock ordering constraints if necessary.
- Patch(es) description should contain:
    - - What is protected
    - - How is protected
    - - Why the current locking scheme has been chosen
    - - Lock ordering constraints if necessary

Hint: Look at `fs/dcache.c` for a good example of locking documentation.

Points

- Maximum points for this assignment are 10.