Operating System: Project 4

Instructed by *Wei Xu*

Due on Jan 8, 2021

Task 2

To make sure that only one thread can access the segment buffer and segment bitmap, we use a mutex lock named segment_lock. Threads intend to enter get_block(), new_data_block(), new_inode_block() and remove_inode() have to acquire segment_lock first and then release it before returning.

To maintain the consistency of the whole file system, we use locks for each independent inode. In each file operation, the thread needs to first locate the inodes it need to access and then acquire the locks corresponding to them.

To make the cache operations atomic, we use a mutex lock for all the cache operations.

To test the concurrency, move testconcurrency.cpp to the mounted disk, run `g++ testconcurrency.cpp -o testconcurrency -std=c++11 -lpthread`. Then run `./testconcurrency 10 10` and open a new terminal. Use htop to see the CPU utility.

The time to compile the whole LFS code in our file system is 7s, where the time cost is 6.3s in ubuntu20.

Task 3

The write-back cache overrides our original `read_block()` and `write_segment()` functions with their `_through_cache()` versions. The cache occupies 4MB space with 512 cachelines, and each cacheline contains contiguous 8 blocks. Upon any cache miss, LRU policy is applied to decide to-be-deleted cachelines. When an eviction finishes, all evicted dirty cachelines are `fsync()`ed to disk.

The cache also provides `init_cache()` and `flush_cache()` functions to support file system initialization and garbage collection.