Operating System: Project 3

Instructed by *Wei Xu*

Due on Dec 13, 2020

Data Structures

Code

Details for Requirements 7-10

- *Requirement 7:* `chmod` and `chown` by modifying metadata in inodes.
- *Requirement 8:* A global lock is declaired in `lfs/utility.cpp`. For simplicity in project 3, all functions require this lock when start and return this lock before return.
- *Requirement 9:* We implemented a segment buffer, and flush it whenever `sync` is called.
- *Requirement 10:* `init` function in `lfs/system.cpp` scans the permanent storage file on disk, and restores the file system before crash. It only takes into account segments which have been successfully written to disk, which can be determined by extra timestamps recorded in each segment. Inode maps can be restored by reading segment summary and imap table in each segment.

Tests

Open terminal in `proj3` directory, then run `bash test.sh`. This will execute all five tests. Refer to `test.sh` for compile options if you want to run each test separately.

`testfile.cpp`

A test for operations `open`, `create` and `write`. This test creates a file with 100 characters. Copy the binary file to an empty directory and execute `./testfile`.

If the file system functions properly, there should be no errors.

`testfile2.cpp`

A test for operations `create`, `write` and `mkdir`, an implicit requirement is thread-safety. This test creates 1000 directories, each with a file inside. Copy the binary file to an empty directory and execute `./testfile2`.

If the file system functions properly, there should be no errors.

`testmkdir.cpp`

A stress test for block-segment management and operation `mkdir`. This test creates directories named $0, 1, 2, \ldots, n-1$. Copy the binary file to an empty directory and execute `./testmkdir <n>`.

If the file system functions properly, there should be no errors.

`testrmdir.cpp`

A stress test for block-segment management and operation `rmdir`. This test creates a tree structure of n directories first, then keeps removing a random directory until all directories are deleted. Copy the binary file to an empty directory and execute `./testrmdir <n>`.

If the file system functions properly, `testrmdir` should not exit due to assertion failure.

`testconcurrency.cpp`

A stress test for block-segment management and thread-safety. This test invokes n threads. Each thread creates m directories, each with a file inside. Copy the binary file to an empty directory and execute `./testrmdir <n> <m>`.

If the file system functions properly, there should be exactly $n \times m$ directories.

Manual

- To compile, execute `scons` in `lfs/` directory. If any issue happens, you may need to use Ubuntu 20.04 and install scons.
- To mount file system, either execute `bash buildfs.sh` in `proj3/` directory, or manually execute (this also applies to the "echo file system" in task 1) `./fuse <mount directory> <options>`.
- Line 190-204 of `lfs/utility.h` contain several debug options. To enable "echo" in log-structured file system (should add `-f` to options), toggle on `DEBUG_PRINT_COMMAND`.
- Line 213 of `lfs/utility.h` is a switch for directory access time updates. When `FUNC_ATIME_DIR` is 1, access timestamps of all files along the path will be updated.

Limitations

We have not implemented garbage collection, so when the file system is full, we can not even delete files.