Extension For An Open Source File System

Jeremy Kato, Skyler Spence, Michael Long, and Jacob Bader

Background - FUSE (Filesystem in USErspace)

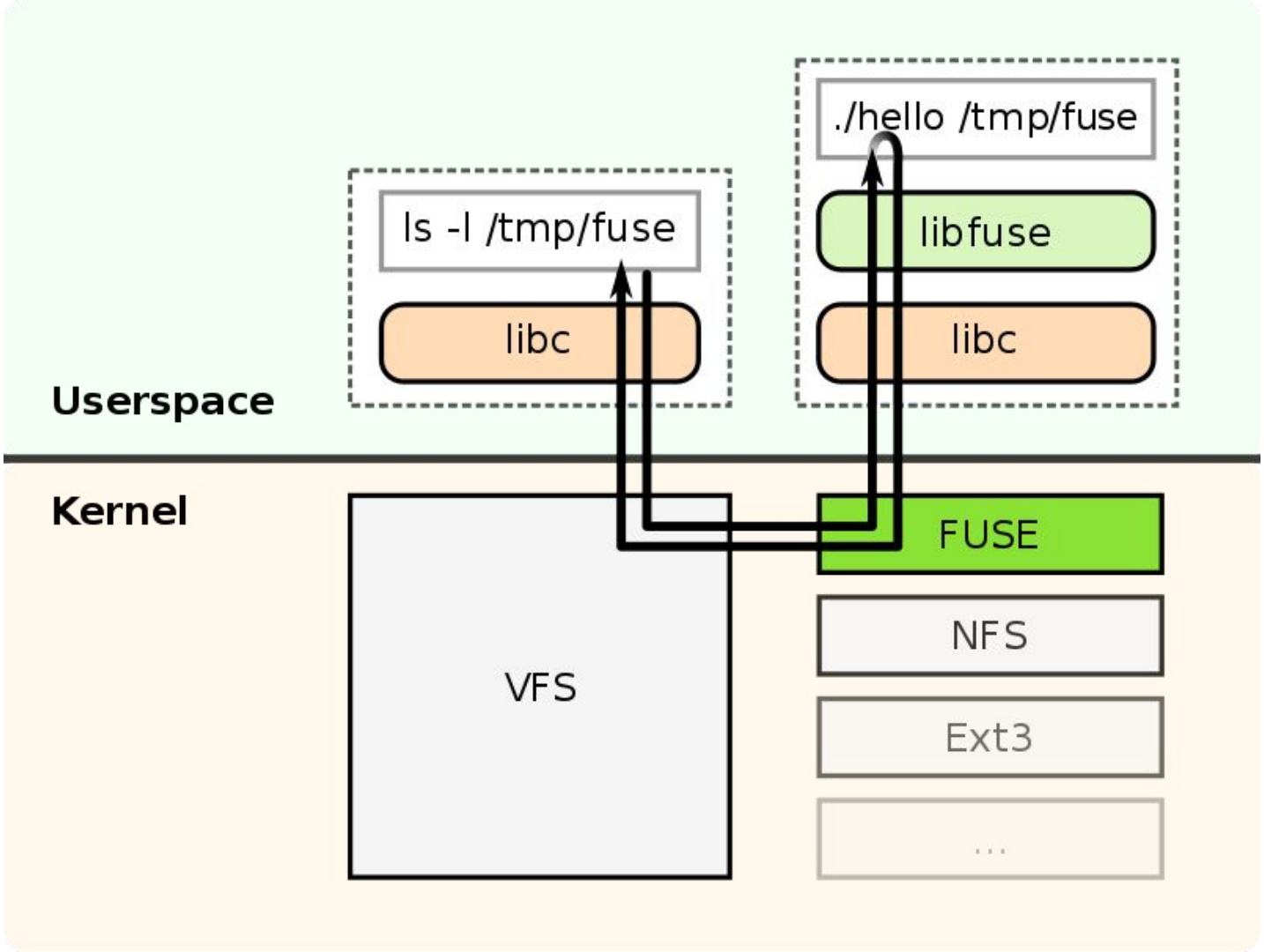
- The main focus of our project is to modify FUSE in order to add new functionality. But what does this mean?

- FUSE is a framework that allows the user to change how files are handled in a space.

Background - FUSE

- Usually, all file operations - reading, writing, etc. - are handled by the operating system. Ordinarily, you can’t easily modify them.

- FUSE allows users to redefine these file operations in order to add or change functionality.


./hello /tmp/fuse
ls -l /tmp/fuse
libfuse
libc
libc
Userspace
Kernel
FUSE
VFS
NFS
Ext3
...

Primary Goals

- We want to develop an extension to FUSE in order to modify the way file operations are handled.
- Our extension will add functionality to the file system.
- Most importantly: to learn about low-level coding and file systems!

New Capabilities

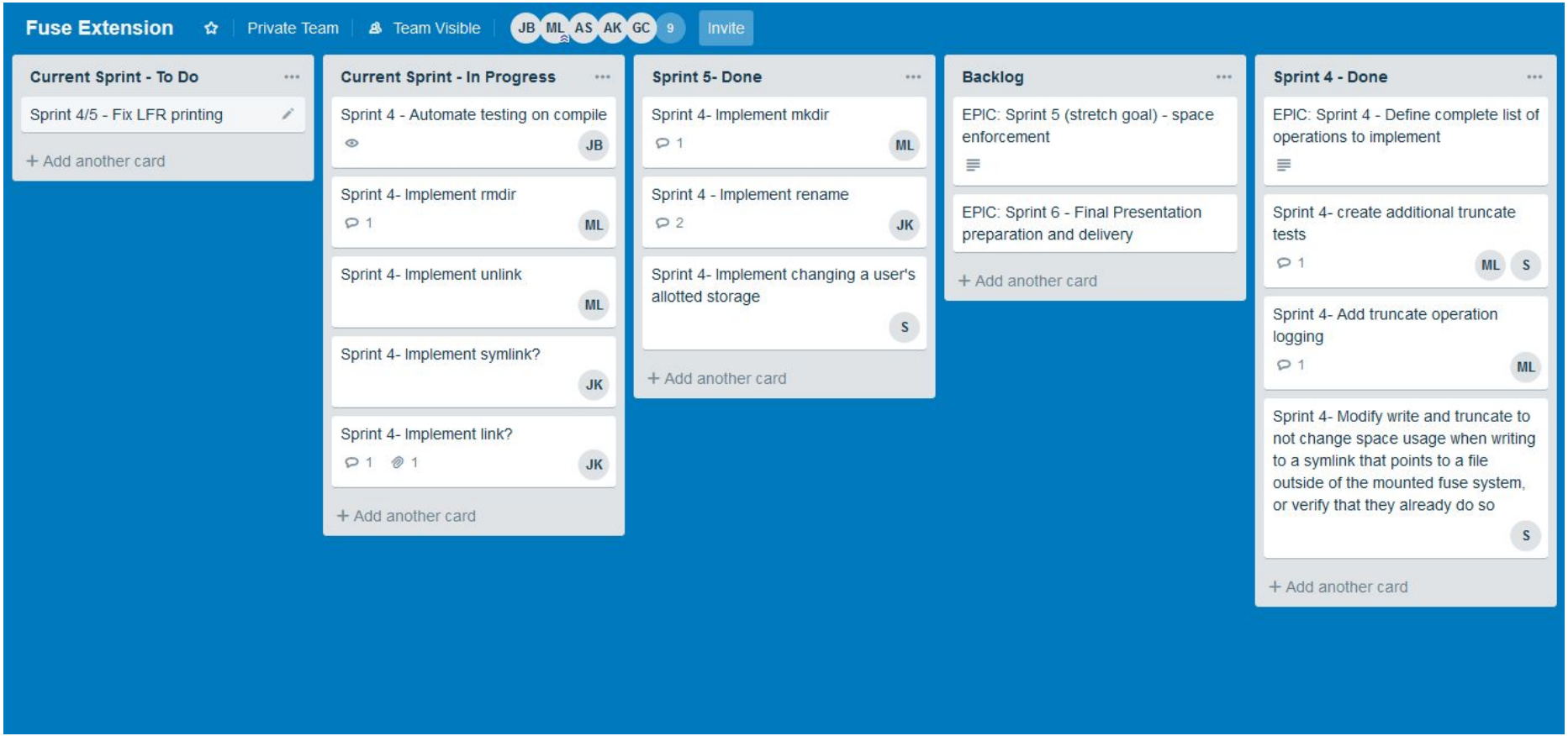
Here are a few things our system does:

- Keep a log of disk space used on a per-user basis
- Keep a log of operations that modify files, such as truncate, write, delete
- Enforce a disk space limit per user, so that there is a maximum limit on space a user can take up in the file system

Other Goals

In addition, we learned about these processes which help us develop in a group environment:

- Keeping objectives clear and work synchronized using software like Trello and Github.
- Using an Agile process to plan out our work.
- Creating automated tests, to ensure that new updates to the code don’t break something that already works.


Fuse Extension
Private Team
Team Visible
JB ML AS AK GC 9
Invite
Current Sprint - To Do
Sprint 4/5 - Fix LFR printing
+ Add another card
Current Sprint - In Progress
Sprint 4 - Automate testing on compile
JB
Sprint 4- Implement rmdir
1
ML
Sprint 4- Implement unlink
ML
Sprint 4- Implement symlink?
JK
Sprint 4- Implement link?
1
1
JK
+ Add another card
Sprint 5- Done
Sprint 4- Implement mkdir
1
ML
Sprint 4 - Implement rename
2
JK
Sprint 4- Implement changing a user's allotted storage
S
+ Add another card
Backlog
EPIC: Sprint 5 (stretch goal) - space enforcement
EPIC: Sprint 6 - Final Presentation preparation and delivery
+ Add another card
Sprint 4 - Done
EPIC: Sprint 4 - Define complete list of operations to implement
Sprint 4- create additional truncate tests
1
ML
S
Sprint 4- Add truncate operation logging
1
ML
Sprint 4- Modify write and truncate to not change space usage when writing to a symlink that points to a file outside of the mounted fuse system, or verify that they already do so
S
+ Add another card

The Database

The Database

- The database is written in Postgresql
  - Went with a database over a flat file due to the ACID properties that the database provides
- The database can be accessed via the lfr program or via normal postgresql queries by locally connecting to libfuse_db
- Consists of 3 Tables:
  - SpaceUsed
  - UserMaxSpace
  - TextLogs

UserMaxSpace

- Our solution has a default maximum space a user is allowed to use
  - 50000 bytes
- Custom space limits can be set by using ./set_space <username> <space_in_bytes>
  - UserMaxSpace holds the entries for all users not using the default

UserMaxSpace

UID	MaxSpace
1000	10000

- Can be accessed via ./lfr -m or ./lfr --max

SpaceUsed

- Both inodes and space written in a file will have an entry in the space used table
- If multiple users have written to a file, a new entry must be made for each user

Space Used

UID	PATH	SpaceUsedInBytes	INODE
1000	/home/user/mounted_dir/directory	4096	true
1000	/home/user/mounted_dir/file	4096	true
1001	/home/user/mounted_dir/file	80000	false

- ./lfr -s or ./lfr --space will sum up all of the space used by each user across all entries and show how much space each user is using in bytes

TextLogs

- Each operation will construct a log string and store it in the TextLog table
- Used mainly for debugging

TextLogs

TextLog
“mknod call. User: 1000 Path: /home/skorve/git-repos/libfuse-branches/libfuse/base_dir/test”
“Write call. User: 1000 Path: /home/skorve/git-repos/libfuse-branches/libfuse/base_dir/test. Size attempted to write: 4. Size actually written: 4. Offset: 0”

- Can be accessed via ./lfr -l or ./lfr --log
  - Will print out the test logs in the order that they were submitted

File Operation Callbacks

File Creation and Deletion Operations

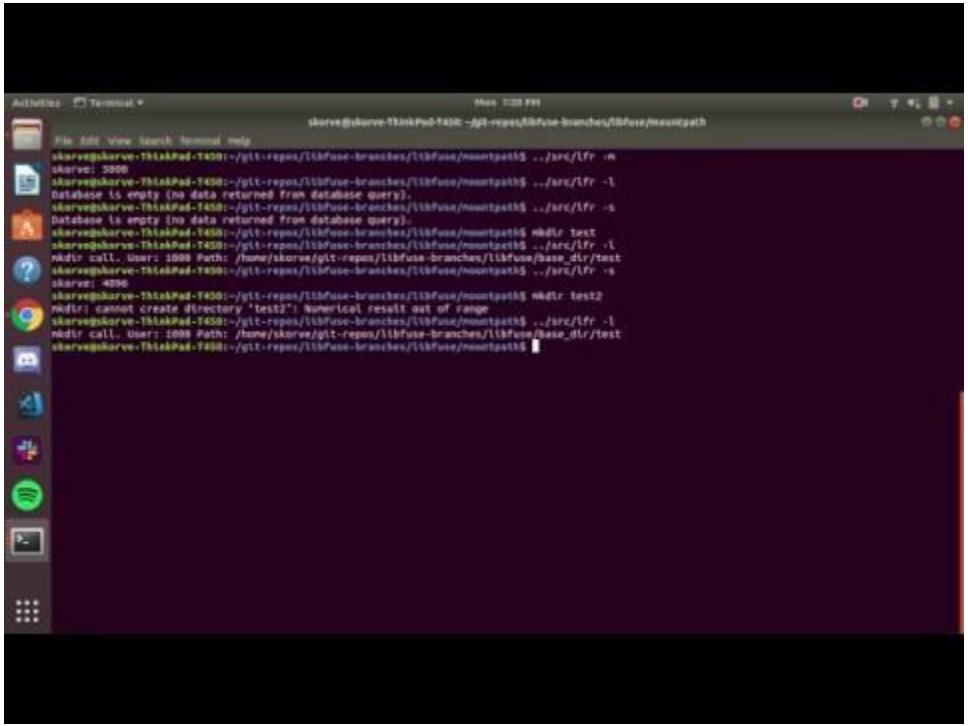
- File creation is handled by mknod
  - Intercept mknod call
  - Check to see if the caller has the space for a new inode
  - If they do call mknod and store the return value
  - If mknod returned with no errors add a new entry in SpaceUsed that is an inode entry with the size of 1 inode and log the operation
- File deletion is handled by unlink
  - Allow the operation to proceed normally and store the return result
  - If the operation did not return an error remove all entries in SpaceUsed that are associated with the file path that unlink was called

File creation and Deletion Operations Demo

Directory Operations

- Directory creation is handled by mkdir
  - 1) Intercept mkdir call
  - 2) Check to see if the caller has the space for a new directory
  - 3) If they do call mkdir and store the return value
  - 4) If mkdir returned with no errors update the database with the new space used and log the operation
- Directory deletion handled by rmdir
  - 1) Allow the operation to proceed normally and store the return result
  - 2) If the operation did not return an error remove the directory entry from the SpaceUsed table and log the operation

Directory Operation Demo



Rename Operation

- 1) Allow rename to execute normally and store the return value
- 2) If rename did not return an error, query the database for records involving the file being renamed.
- 3) Update the database records by replacing any rows with the old filename with the new filename.

Rename Demo

Write Operation

Appending is easy.

But also needs to deal freeing space usage from:

- Overwriting data you wrote
- Overwriting data another user wrote
- Overwriting data in a file multiple users wrote to
- Rounding errors

Write Operation: Overwriting Data in a File Multiple Users Wrote to

- User A creates file f and writes 10 MB to it
- User B appends 10 MB to to f
- User C overwrites 10 MB in f
- Users A and B should both get (10 / 20) * 10 = 5 MB freed

Write Operation: Rounding Errors

- Floating point arithmetic has limited precision
- Makes sure that if you overwrite n bytes, the total space freed by users is exactly n
- Loops through every user who wrote to the file
    - Keeps track of space left to distribute
    - Makes sure space left to distribute is exactly 0 once every user has gotten space distributed to it

Write Demo

Truncate Operation

- Can actually increase file size when truncating
- Deals with redistributing space freed similarly to how write does so
- Only operates on non-inode space entries

Truncate Demo

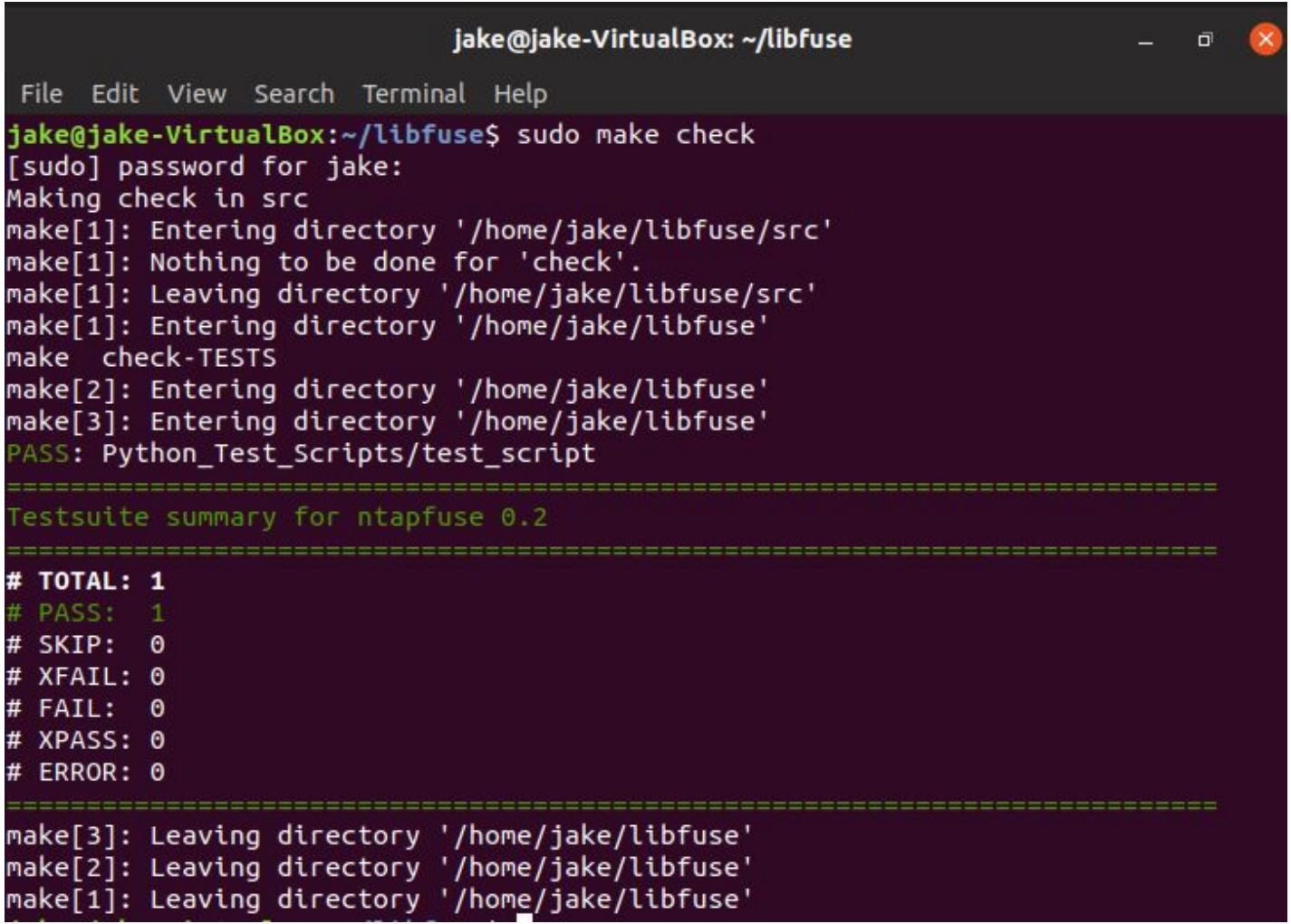
Testing

Unit Tests

- Written in python using the unittest framework
- Makes use of subprocess.Popen().communicate() to execute command line commands
    - Opens a new subprocess with a passed in command line commands and then waits for it to terminate
    - Mkdir, rmdir, echo, rm etc
    - Use lfr commands to observe results in the database
- Writes results to a text file

Build/Testing Process

- Run autoreconf --install and ./configuren
- Run ‘make check’ - this compiles the code and runs the unit tests.
- Unit tests take about 2-3 minutes to run, and output whether the tests pass.


jake@jake-VirtualBox: ~/libfuse
File Edit View Search Terminal Help
jake@jake-VirtualBox:~/libfuse$ sudo make check
[sudo] password for jake:
Making check in src
make[1]: Entering directory '/home/jake/libfuse/src'
make[1]: Nothing to be done for 'check'.
make[1]: Leaving directory '/home/jake/libfuse/src'
make[1]: Entering directory '/home/jake/libfuse'
make check-TESTS
make[2]: Entering directory '/home/jake/libfuse'
make[3]: Entering directory '/home/jake/libfuse'
PASS: Python_Test_Scripts/test_script
Testsuite summary for ntapfuse 0.2
TOTAL: 1
PASS: 1
SKIP: 0
XFAIL: 0
FAIL: 0
XPASS: 0
ERROR: 0
make[3]: Leaving directory '/home/jake/libfuse'
make[2]: Leaving directory '/home/jake/libfuse'
make[1]: Leaving directory '/home/jake/libfuse'

Example Unit Test Results

test_display_deletions (display_file_IO_logs_test.case_display_deletions) ... ok
test_display_mkdir (display_file_IO_logs_test.case_display_mkdir) ... ok
test_display_truncates (display_file_IO_logs_test.case_display_truncates) ... ok
test_display_writes (display_file_IO_logs_test.case_display_writes) ... ok
test_one_user_no_write (display_space_usage_test.case_one_user_no_write) ... ok
test_one_user_write (display_space_usage_test.case_one_user_write) ... ok
test_two_users_write_to_same_file (display_space_usage_test.case_two_users_write_to_same_file) ... ok
test_two_users_write_to_separate_files (display_space_usage_test.case_two_users_write_to_separate_files) ... ok
test_append_to_existing_file (record_space_usage_test.case_append_to_existing_file) ... ok
test_case_delete_existing_file (record_space_usage_test.case_delete_existing_file) ... ok
test_delete_file_two_writers (record_space_usage_test.case_delete_file_two_writers) ... ok
test_make_new_directory (record_space_usage_test.case_make_new_directory) ... ok
test_case_modify_existing_file (record_space_usage_test.case_modify_existing_file) ... ok
test_truncate_beyond_user_written_in_file_two_writers (record_space_usage_test.case_truncate_beyond_user_written_in_file) ... ok
test_case_truncate_existing_file (record_space_usage_test.case_truncate_existing_file) ... ok
test_write_over_existing_file (record_space_usage_test.case_write_over_existing_file) ... ok
test_case_write_to_new_file (record_space_usage_test.case_write_to_new_file) ... ok
test_case_create_and_rename_file (rename_file_test.case_rename_existing_file) ... ok
test_case_rename_nonexistent_file (rename_file_test.case_rename_nonexistent_file) ... ok

---

Ran 19 tests in 136.477s

OK

Conclusions

Next Steps

- Clean up redundant logging code
- Expand mknod to allow other users to own file inodes even if another user owns the mounted system
- Add transactional locking when processing file requests
- Investigate link and symlink for implementation

Challenges

- Issues connecting the database to fuse
- Vm issues
- Balancing with other course work

What We Learned

- Became much more familiar working in a linux environment
- Best practices when using GitHub for version control
- Using the Agile Process to stay organized
- How to organize large requirements into smaller tasks
- How to evolve the team’s development process to make development more efficient

Questions