P1:Client/Server File System

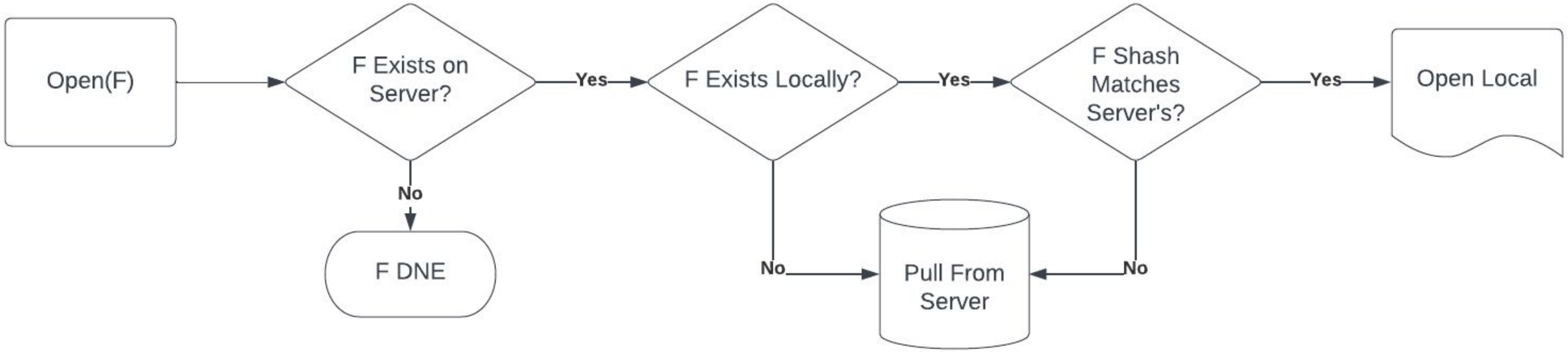
Evan Wireman
Mahitha Pillodi
Ruthvika Reddy Loka

Design

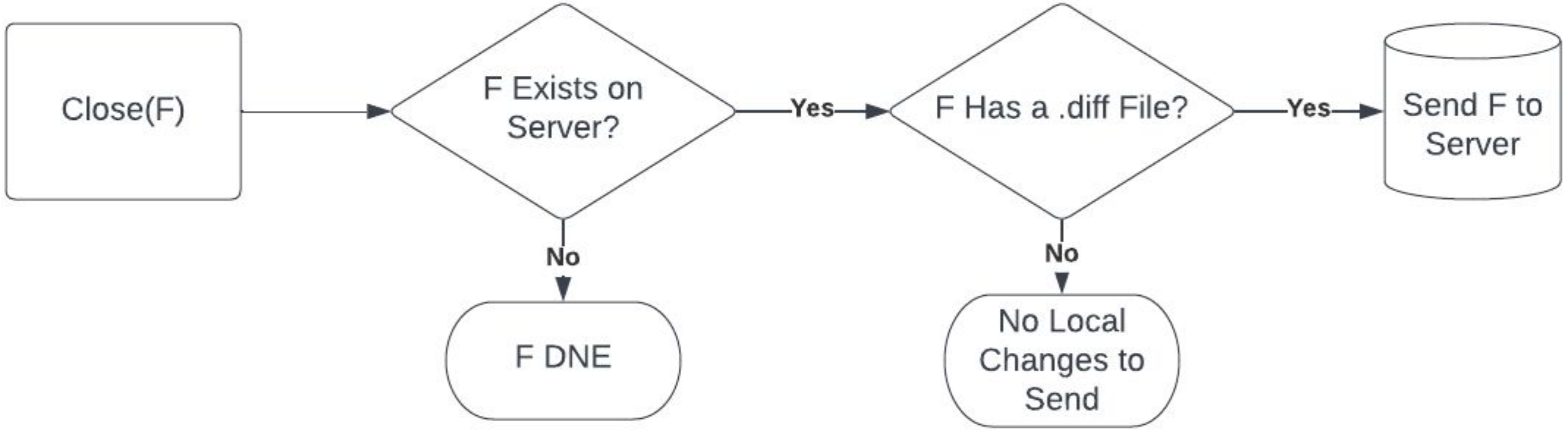
Defining Structural Principles

- Hash of the *struct stat* of server's latest version of a file acts as uniquifier (we will call this a file's shash)
  - When client opens a file that exists on the server, the client pulls the file and its shash
  - Iff the client closes and flushes to the server, the server updates the file's shash
  - Both client and server maintain a mapping of filenames to sashes
- On flush, client makes a *.diff* file, representing that the local version of a file has been changed
  - This prevents unnecessary server writes if the file was just read from and never modified
- Rename, mkdir, rmdir, and mknod all communicate with the server
  - This was done to make the server mapping of paths to shashes easier to maintain

Open

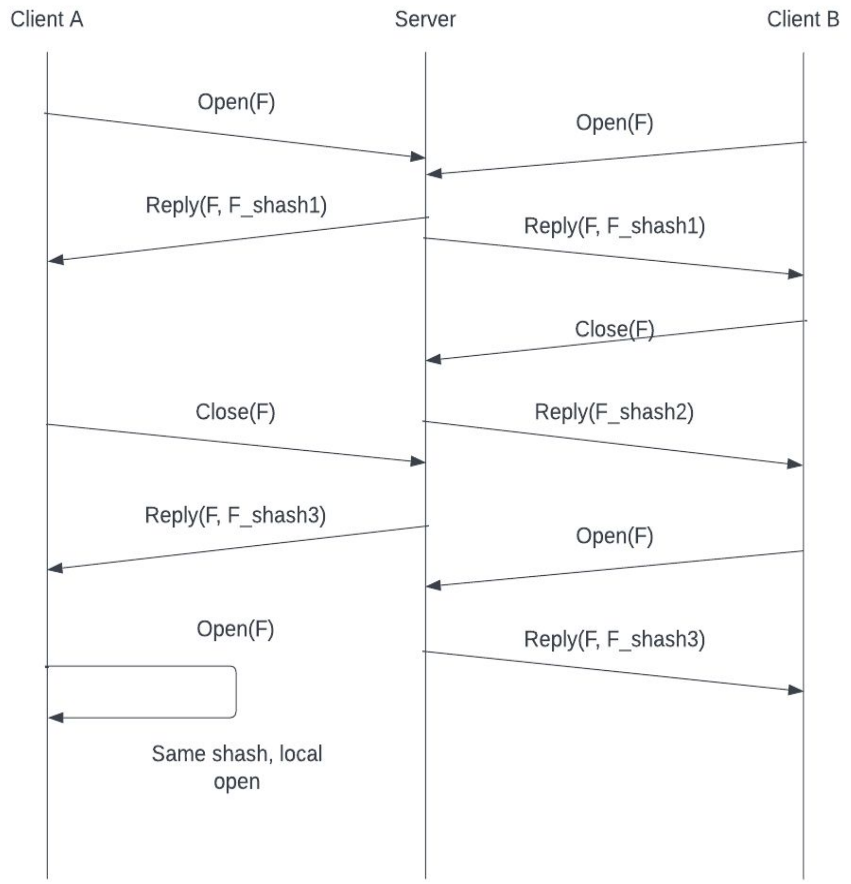

Open(F)
F Exists on Server?
Yes
No
F DNE
F Exists Locally?
Yes
No
F Shash Matches Server's?
Yes
No
Pull From Server
Open Local

Close


Close(F)
F Exists on Server?
Yes
No
F DNE
F Has a .diff File?
Yes
No
No Local Changes to Send
Send F to Server

Shash semantics

- While we are still experiencing a few small kinks with the system, this was the intended approach
- This strategy maintains last-writer-win consistency
- On client crash, its cache is empty, so it will pull all opens from the server



Misc. Design Choices

- RPC payload size of 2mb
  - This only came into play when streaming files
  - Maximum RPC payload size is 4mb, looking back this was quite conservative
- unlink is treated as delete
  - Since we are not concerned about linking, we figured it was safe to treat all unlinks as deletes

Durability and Reliability

Durability: Client Crash Consistency Protocol

- Client crash
  - Upon reboot, it will have no shashes saved. Thus, any files it opens will be requested from the server
  - Any files that were opened prior to the crash but were not closed will be treated as dirty
  - **Consistency Issue:** If the client has a .diff file saved (i.e. it flushed but was unable to close), the local changes will not be conveyed to the server upon reboot.
- Server crash
  - Whenever the server responds to a client's GetHash request, it checks if it has a shash saved. If not, it checks if there is a local file, and if so, generates a shash for it and saves the mapping.
  - This causes all clients to have a stale shashes, since *struct stat* tracks last file access. Thus, any subsequent opens from clients will pull from the server

Performance

Performance testing setup

- All filebench benchmarks were found running our system on a single machine
  - The cloudlab machines were producing weird outputs regarding the number of operations performed
  - The total runtime and ms/op were quite similar for a majority of the workloads, so we agreed that testing in this manner would not discredit our measurements
- Our implementation had incredibly slow Create/Delete times
  - While all of the workloads ran, some of the filebench workloads that created/removed a large number of files took incredibly long to preallocate
  - We adapted any workload that worked on 2000+ files to instead generate 2000 files
  - From what we could tell, this did not dramatically change the ms/op measurement

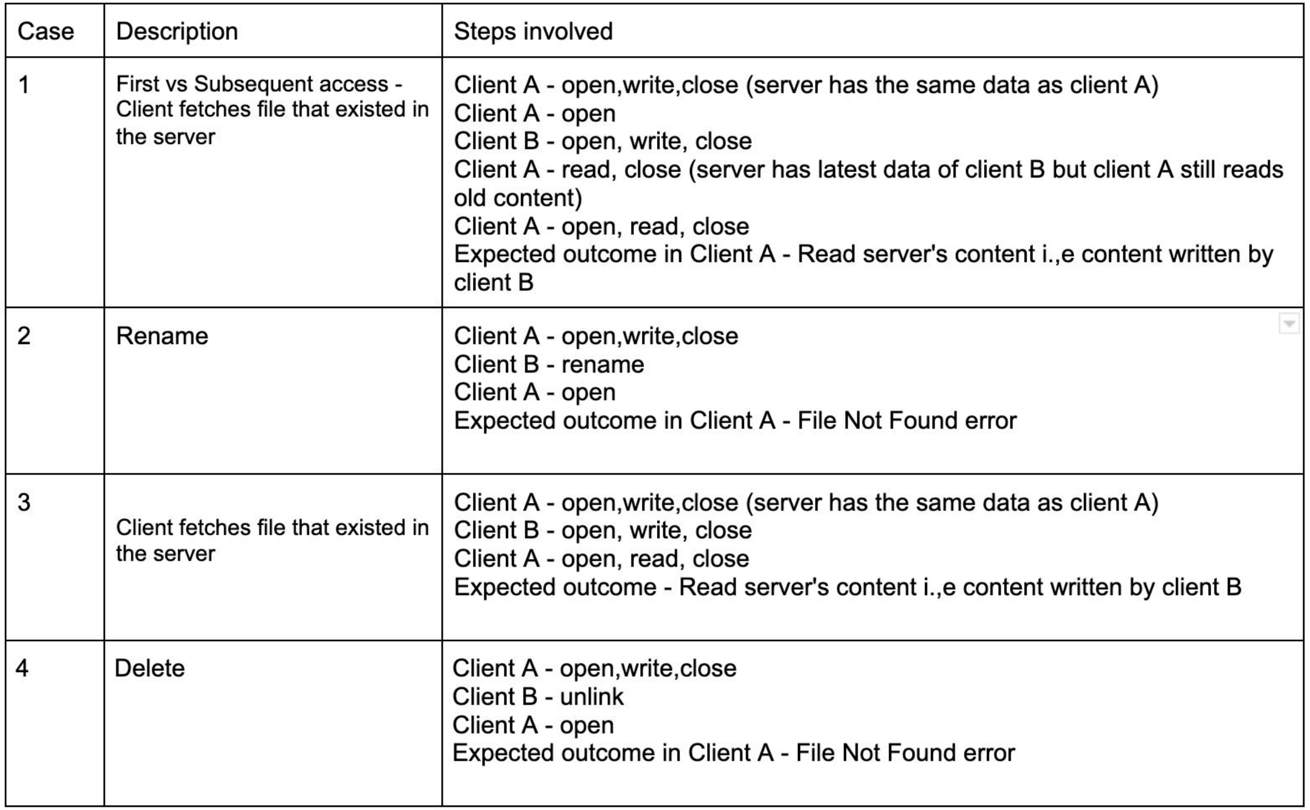
Filebench workloads

Workload	ms/op
filemicro_create.f	21.702
filemicro_createfiles.f	30.213
filemicro_createrand.f	24.769
filemicro_delete.f	345.662
filemicro_read.f	0.125
filemicro_rwritedsync.f	0.634
filemicro_seqread.f	0.146
filemicro_seqwrite.f	41.311
filemicro_statfile.f	160.623
filemicro_writefsync.f	25.965
fileserver.f	**Failed**
mongo.f	24.231
varmail.f	138.642
webserver.f	244.642

Potential Improvements

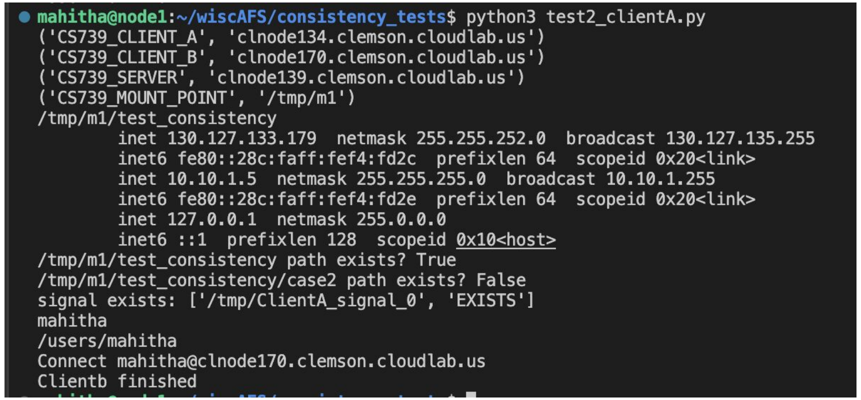
- Not communicating with server on mknod would improve performance
  - FUSE create runs mknod followed by open
  - Devising a manner for the client to be aware of files that were created and not yet opened would prevent several RPC calls
  - Once the file is closed, the new file could be written to the server
- Fully leveraging RPC payload size
  - We used payload lengths of 2mb. This was rather conservative, since the maximum payload size is 4mb
  - This means we are making far more RPC calls than necessary when transmitting files, likely impacting our performance
- We perform a lot of stat calls when generating sashes
  - The design choice to use shashes implies frequent stat calls on the server. This likely slows down performance
  - Perhaps storing different information as a uniquifier would have improved performance

Consistency Test Scenarios

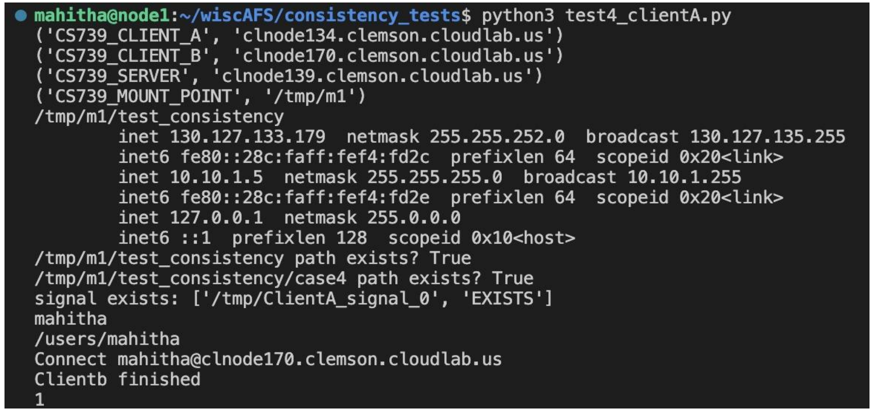


Case	Description	Steps involved
1	First vs Subsequent access - Client fetches file that existed in the server	Client A - open,write,close (server has the same data as client A) Client A - open Client B - open, write, close Client A - read, close (server has latest data of client B but client A still reads old content) Client A - open, read, close Expected outcome in Client A - Read server's content i.,e content written by client B
2	Rename	Client A - open,write,close Client B - rename Client A - open Expected outcome in Client A - File Not Found error
3	Client fetches file that existed in the server	Client A - open,write,close (server has the same data as client A) Client B - open, write, close Client A - open, read, close Expected outcome - Read server's content i.,e content written by client B
4	Delete	Client A - open,write,close Client B - unlink Client A - open Expected outcome in Client A - File Not Found error

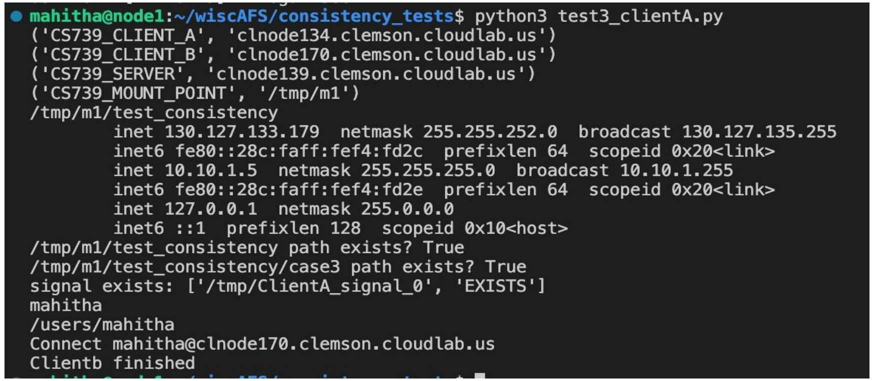
Output



```
mahitha@node1:~/wiscAFS/consistency_tests$ python3 test2_clientA.py
('CS739_CLIENT_A', 'clnode134.clemson.cloudlab.us')
('CS739_CLIENT_B', 'clnode170.clemson.cloudlab.us')
('CS739_SERVER', 'clnode139.clemson.cloudlab.us')
('CS739_MOUNT_POINT', '/tmp/m1')
/tmp/m1/test_consistency
        inet 130.127.133.179  netmask 255.255.252.0  broadcast 130.127.135.255
        inet6 fe80::28c:faff:fef4:fd2c  prefixlen 64  scopeid 0x20<link>
        inet 10.10.1.5  netmask 255.255.255.0  broadcast 10.10.1.255
        inet6 fe80::28c:faff:fef4:fd2e  prefixlen 64  scopeid 0x20<link>
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
/tmp/m1/test_consistency path exists? True
/tmp/m1/test_consistency/case2 path exists? False
signal exists: ['/tmp/ClientA_signal_0', 'EXISTS']
mahitha
/users/mahitha
Connect mahitha@clnode170.clemson.cloudlab.us
Clientb finished
```



```
mahitha@node1:~/wiscAFS/consistency_tests$ python3 test4_clientA.py
('CS739_CLIENT_A', 'clnode134.clemson.cloudlab.us')
('CS739_CLIENT_B', 'clnode170.clemson.cloudlab.us')
('CS739_SERVER', 'clnode139.clemson.cloudlab.us')
('CS739_MOUNT_POINT', '/tmp/m1')
/tmp/m1/test_consistency
        inet 130.127.133.179  netmask 255.255.252.0  broadcast 130.127.135.255
        inet6 fe80::28c:faff:fef4:fd2c  prefixlen 64  scopeid 0x20<link>
        inet 10.10.1.5  netmask 255.255.255.0  broadcast 10.10.1.255
        inet6 fe80::28c:faff:fef4:fd2e  prefixlen 64  scopeid 0x20<link>
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
/tmp/m1/test_consistency path exists? True
/tmp/m1/test_consistency/case4 path exists? True
signal exists: ['/tmp/ClientA_signal_0', 'EXISTS']
mahitha
/users/mahitha
Connect mahitha@clnode170.clemson.cloudlab.us
Clientb finished
1
```

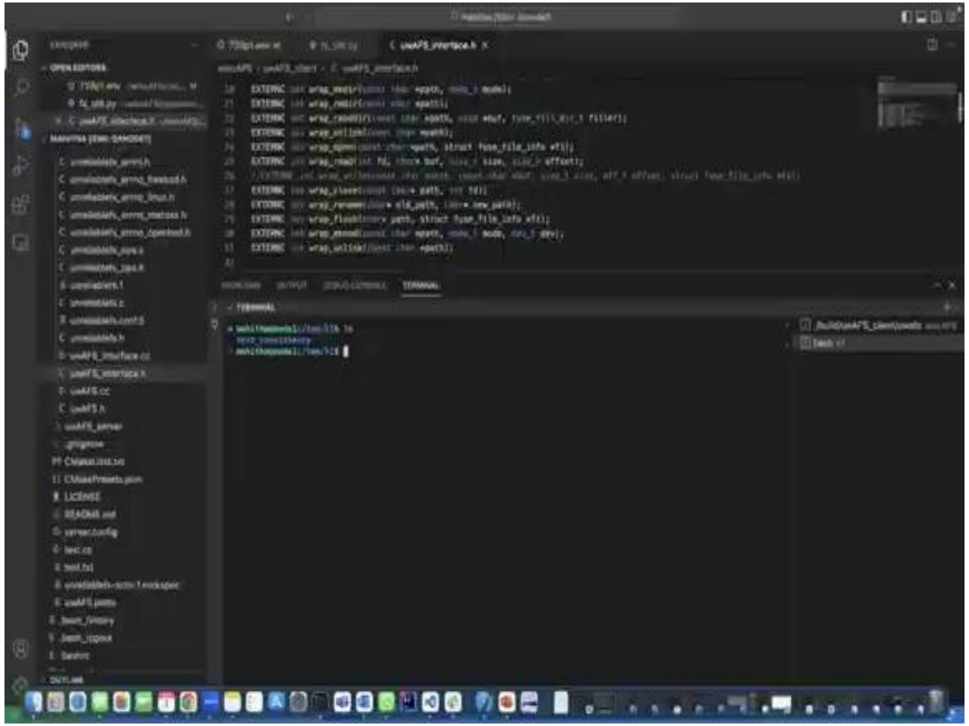


```
mahitha@node1:~/wiscAFS/consistency_tests$ python3 test3_clientA.py
('CS739_CLIENT_A', 'clnode134.clemson.cloudlab.us')
('CS739_CLIENT_B', 'clnode170.clemson.cloudlab.us')
('CS739_SERVER', 'clnode139.clemson.cloudlab.us')
('CS739_MOUNT_POINT', '/tmp/m1')
/tmp/m1/test_consistency
        inet 130.127.133.179  netmask 255.255.252.0  broadcast 130.127.135.255
        inet6 fe80::28c:faff:fef4:fd2c  prefixlen 64  scopeid 0x20<link>
        inet 10.10.1.5  netmask 255.255.255.0  broadcast 10.10.1.255
        inet6 fe80::28c:faff:fef4:fd2e  prefixlen 64  scopeid 0x20<link>
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
/tmp/m1/test_consistency path exists? True
/tmp/m1/test_consistency/case3 path exists? True
signal exists: ['/tmp/ClientA_signal_0', 'EXISTS']
mahitha
/users/mahitha
Connect mahitha@clnode170.clemson.cloudlab.us
Clientb finished
```

Demos

Filebench

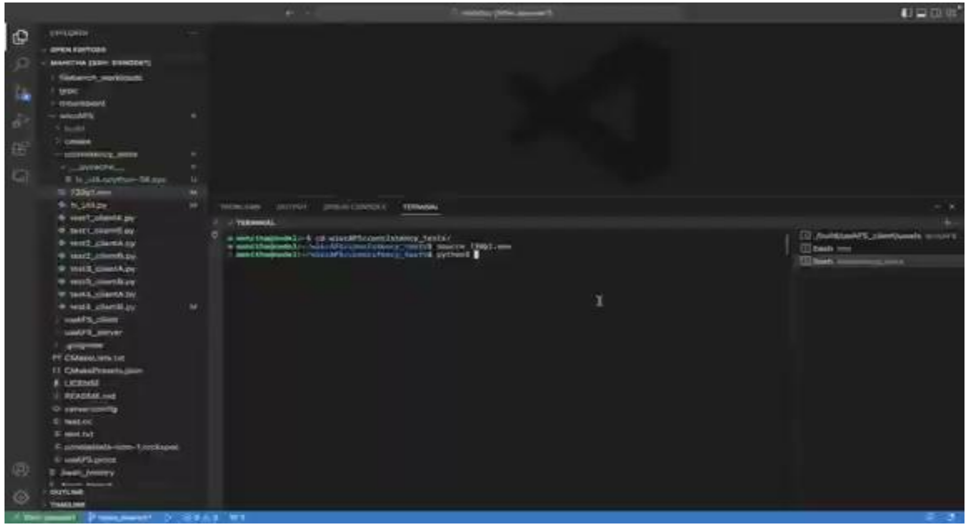
Basic functionality



```
EXTERNC int wrap_mkdir(const char *path, mode_t mode);
EXTERNC int wrap_rmdir(const char *path);
EXTERNC int wrap_readdir(const char *path, void *buf, fuse_fill_dir_t filler);
EXTERNC int wrap_unlink(const char *path);
EXTERNC int wrap_open(const char *path, struct fuse_file_info *fi);
EXTERNC int wrap_read(int fd, char* buf, size_t size, size_t offset);
//EXTERNC int wrap_write(const char *path, const char *buf, size_t size, off_t offset, struct fuse_file_info *fi);
EXTERNC int wrap_close(const char* path, int fd);
EXTERNC int wrap_rename(char* old_path, char* new_path);
EXTERNC int wrap_flush(char* path, struct fuse_file_info *fi);
EXTERNC int wrap_mknod(const char *path, mode_t mode, dev_t dev);
EXTERNC int wrap_unlink(const char *path);
```

```
[illegible] ls
test_consistency
[illegible] █
```

Consistency tests



```
[illegible]:~$ cd wiscAFS/consistency_tests/
[illegible]:~/wiscAFS/consistency_tests$ source [illegible]
[illegible]:~/wiscAFS/consistency_tests$ python3
```

Thank you!