Assignment 3 System Manual

Name: Ze Ran Lu Student Number: 20578889

Implementation

The program is designed in the same way as the specification. The client mantains a mapping of path to file handler to avoid double open and the server maintains a mapping, protected by a mutex, of path to an entries, which contains the read-write lock for this file, the number of write openers and the number of read openers. In this way, the server will know if a file is already opened in write mode, or when it is time to clean up the entry.

The calls rely on two procedures that does the transfer:

```
int push(const char *path, const char *cache_full_path, off_t size_client, struct
timespec *ts)
```

The process is:

```
# local syscalls to read from local cache
open # O_RDONLY
read # read to buffer
close

# A2 calls to write to server
A2::watdfs_cli_open # with O_CREAT | O_WRONLY | O_TRUNC
A2::watdfs_cli_write # write to buffer
A2::watdfs_cli_utimens # change remote timestamp
A2::watdfs_cli_release
```

This will upload the file along with some metadata, and will return `-ENOENT` if `path` does not exists in the cache.

```
int pull(const char *path, const char *cache_full_path, off_t size_server, struct
timespec *ts)
```

The process is:

```
# A2 calls to read from server
A2::watdfs_cli_open # O_RDONLY
A2::watdfs_cli_read # read to buffer
A2::watdfs_cli_release

# local syscalls to create local cache
open # with O_CREAT | O_RDWR | O_TRUNC
```

```
write # write from buffer
fsync # flush to the local cache
futimens # change the local timestamp
close
```

This will download the file and assign a metadata, and will return `-ENOENT` if `path` does not exists in the server.

Of course, these procedures are used as needed by checking the freshness condition for some of the calls.

To check freshness condition:

```
stat # local stat call

# if the cache interval is exceeded, get stat from server to do further check
A2::watdfs_cli_getattr # A2 call to get stat from server
# then compare the timestamps
```

The `push` and `pull` procedures are protected by two RPC calls to guarantee the atomicity of transfer in both ways:

```
int lock(const char *path, mode_t mode)
```

```
int unlock(const char *path, mode_t mode)
```

These procedures will return `-ENOENT` if entry does not exists in the server mapping, `-EINVAL` if the mode is not a valid mode, or `-errno`.

For calls that does not require `watdfs_cli_open` such as `watdfs_cli_mknod`, it uploads the file after the local syscall when freshness condition holds.

A3 Calls Return Codes

- watdfs_cli_open returns `-EMFILE` if the file is already opened in the client and `-EACCES` if the file is already opened in the server.

Testing

Both client and server code can have `CHECK_ERROR` macro enabled. It will return the return code and `errno` if the return code is less than 0, and print the current filename, function name and line number. This is useful in testing.

There are some test programs located in `tests` folder. They were compiled and placed in the parent of `mount` folder to test syscall individually.

- opentest.c: tests if open and close works correctly and if the file is created in the server.
- readtest.c: tests if read works correctly.
- writetests.c: tests if write works correctly.
- truncatetest.c: tests if truncates extends or truncates a file to 1000 bytes.
- doubleopenwwtest.c: tests if `-EACCES` is returned if two clients opens the same file in write mode.
- doubleopenwrtest.c: tests concurrent read and write by two clients.
- doubleopenrrtest.c: tests concurrent reads by two clients.

To run two clients, run `client.sh` and `client_second.sh`. They will create two mount and cache folders.

The program has been also tested manually (to check if timestamp works correctly) by using commands like:

```
$ touch mount/a # create a file 'a' or update the modified time if the file already
exists
$ stat mount/a cache/a server/a # check timestamps works correctly
$ yes test | head -10000 > mount/a # create a file of 10000 lines of 'test'
$ sha1sum mount/a server/a cache/a # check if the file integrity
```

Four cases are also tested manually: file in both cache and server, file in cache only, file in server only, file not in both folders.