Operating System: Project 3

Instructed by *Wei Xu*

Due on Dec 13, 2020

Data Structures

Code

Details for Requirements 7-10

Tests

Open terminal in `proj3` directory, then run `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.

Command-line Tests

Open a shell in directory `lfs`, and execute `./fuse disk100Mi` first. Then you are free to try any of the following command-line tests. To deal with file name conflicts between tests, you may directly use `rm -rf *` to wipe LFS. These tests are based on Linux shell commands, so the correct results can be obtained by trying on a real Linux system (however, updates for `atime` may be slightly different).

Note: *due to the implementation of FUSE, commands are executed under the permission of* `others`*. This should be dealt with caution when analyzing the results of the following tests.*

Test for permission control Run through the following commands to test permission control of files and directories. Use `chmod` to change permission. Directory should contain some files initially.

(1) Files. Under permission `774`, file is readable but not writable; under permission `776`, file is both readable and writable. It is trickier to test for `772` (file is writable but not readable), and you have to write a simple C++ program. ***Note:*** *file permission is* `664` *by default, so we manually run* `chmod 666` *below.*

(2) Directories. Under permission `774`, `772` and `771`, the directory (a) can only be read (e.g. `ls a`), write (e.g. `touch a/f.txt`) and accessed (e.g. `cd a`), respectively. Permissions are composable.

Note: *you may disable permission by flags* `ENABLE_PERMISSION` *(for internal control by internal "**if**"s) and* `ENABLE_ACCESS_PERM` *(for external permission queries through* `access`*), since they follow different mechanisms. You may refer to the manual below.*

Test for timestamps Run through the following commands in the first column of the table.

Commands	`stat ?`	no flags	`nodiratime`	`nodiratime & relatime`
`mkdir a`	a	a, m, c are initialized to the same.		
`touch a/x.txt`	a	a, m, c	m, c	a, m, c
`ls a`	a	a, c	—	—
`mv a b`	b	c	c	c
`ls b`	b	a, c	—	a, c
`chmod 666 b/x.txt`	x.txt	c	c	c
`echo "abc" >> b/x.txt`	x.txt	a, m, c	a, m, c	a, m, c
`cat b/x.txt`	x.txt	a, c	a, c	—
`mv b/x.txt b/y.txt`	y.txt	c	c	c
`cat b/y.txt`	y.txt	a, c	a, c	a, c

Manual

Bugs