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.

Manual

Bugs