Himanshu Mehru

CSS430, Spring 2014

6/11/2014

Final Project Report

# Design Specification:

Since we initially wrote the files individually, we came together and wrote the design specification as a group.

## Directory.java:

For this file we were given a set of methods and we had to implement them. Bytes2directory was given to us from the final project PDF while everything else we had to create. The directory2bytes method was essentially the same thing as the bytes2directory just reversed. Whereas the ialloc, ifree and namei we had to put a little more work into. Ialloc created a new inode number for the given filename. So we used a loop to check for an empty space to put the filename and inumber into the file size array. Ifree looks for the iNumber in the fsize array and gets rid of it, and finally namei returns the iNumber corresponding to the filename by searching through the array and doing a string comparison. If it finds it will return an iNumber as a short.

## Inode.java:

The inode file was a pain and doesn’t function 100% properly. The purpose of the inode is that it describes one file. Part of the Inode constructor was given to us, the part where we can save the data into the variables. However what we added was to create the new iNode object with the new values by using a for loop and saving the information into the direct array. We also saved information into the indirect block. The toDisk method does almost the same thing as the constructor except it’s in reverse and uses rawwrite to the disk to save the information. Then we have the getIndexBlockNumber method which just returns the indirect block. The setIndexBlock method that does error checking and saves the index block number into the indirect block. We have getBlockId which just returns the appropriate blockID number by calculating it with the iNumber. Then we have the setBlock method that takes the iNumber and blockId, checks in the direct array for empty space, if there is empty space, then go ahead and save it in there. If there isn’t then look in the indirect and save it in there, if you can’t, then return false because we couldn’t save the information. Finally, we have the findTargetBlock, which does a great job when you don’t want to append, it takes the offset and divides it by the total blocksize and searches for the block in the direct array. However, when it comes to append, we had errors that we couldn’t solve.

## FileSystem.java:

The FileSystem.java handles all reads, writes, and appends to a file. It is the lowest level for these commands. A write call is made which is sent through SysLib, to the kernel, and then back to FileSystem. The constructor was given to us to set up the different variables so we implemented sync, open, close, fsize, read, write, deallocAllBlocks, delete, and seek. Sync, syncs to the disk and writes the contents of directory entry to byte array. Format formats the filesystem, open opens a file with a filename and a mode to go with it (either w, w+, r, a). Close will close a file with the FileTableEntry and call deallocAllBocks which deallocated all the blocks from memory. The method fsize just gets the file size from the inode within the FileTableEntry. Read will read a file, write will write a file, delete will delete a file with a certain filename and seek will move the seek pointer to the appropriate location.

## FileTable.java:

A structure that helps maintain the different files between threads. We were given the constructor as well as the methods needed to complete the file, so we had to fill in the blanks with code. For falloc we allocate a new file and return a reference to that file table entry. With ffree we free a file table entry, and fempty checks to see if the table is empty. All of these methods are synchronized which allows the multithreading to occur.

## FileTableEntry.java:

For this file, we made some modifications to make error checking much easier. We converted the “mode” value from a string to an integer. We also created a new function called convertModeToShort(String m) that took a string, forced it to lower case, and did a check very similar to the constructor check where we compared the string to a letter and returned an integer value depending on what it was. This allowed us to be able to always check and refer to a number instead of having to compare the string over and over again.

## SuperBlock.java:

The SuperBlock file is used to describe the number of disk blocks, the number if inodes and the block number of the head blocks of the free list. We were given the constructor in the final pdf file, but we also implemented format, sync, getFreeBlock, and returnBlock. Format determines how many blocks should be allocated for Inodes and where the free list should start. Sync writes back totalBlocks, totalInodes and freeList to the disk. The method getFreeBlock dequeues the top block from the free list and returnBlock enqueues a given block to the free list.

## TCB.java:

All we had to do is add a for loop that creates a new FileTableEntry called ftEnt and fills all of its entries to null.

# Test Results:

-->l Test5

l Test5

threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)

1: format( 48 )...................successfully completed

Correct behavior of format......................2

2: fd = open( "css430", "w+" )....successfully completed

Correct behavior of open........................2

3: size = write( fd, buf[16] )....successfully completed

Correct behavior of writing a few bytes.........2

4: close( fd )....................successfully completed

Correct behavior of close.......................2

5: reopen and read from "css430"..successfully completed

Correct behavior of reading a few bytes.........2

6: append buf[32] to "css430".....successfully completed

Correct behavior of appending a few bytes.......1

7: seek and read from "css430"....successfully completed

Correct behavior of seeking in a small file.....1

8: open "css430" with w+..........successfully completed

Correct behavior of read/writing a small file.0.5

9: fd = open( "bothell", "w" )....successfully completed

10: size = write( fd, buf[6656] ).successfully completed

Correct behavior of writing a lot of bytes....0.5

11: close( fd )....................successfully completed

12: reopen and read from "bothell"successfully completed

Correct behavior of reading a lot of bytes....0.5

13: append buf[32] to "bothell"...buf[0] = 16 buf6656 = 0

14: seek and read from "bothell"...seek(fd,512\*11,0) contents 16(wrong)

Should be: 0

15: open "bothell" with w+.........tmpBuf[0]=21 (wrong)

Should be : 0

16: delete("css430")..............successfully completed

Correct behavior of delete....................0.5

17: create uwb0-29 of 512\*13......successfully completed

Correct behavior of creating over 40 files ...0.5

18: uwb0 read b/w Test5 & Test6...

threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)

Test6.java: fd = 3successfully completed

Correct behavior of parent/child reading the file...0.5

19: uwb1 written by Test6.java...Test6.java terminated

Correct behavior of two fds to the same file..0.5

Test completed

# Performance and Issues:

The program appears to have a fast performance, and does most of the required file operations. However, according to Test5, test cases 13, 14, and 15 do not work correctly. We have spent the last several days trying to fix this bug, and we still haven’t figured out the solution. We believe that it occurs in the findTargetBlock method in Inode.java, but we couldn’t manage to fix this error.

# Contributions:

I worked on creating FileTable.java and SuperBlock.java, as well as help test and debug the file system once all the code came together. I did about 1/3 of the work.

Elliot worked on creating FileSystem.java, and most of the testing and debugging. He was responsible for fixing most of our errors. Elliot did about 1/3 of the work.

Will worked on creating TCB.java, Inode.java, and Directory.java. He did testing and debugging as well. He did about 1/3 of the work.

Overall, everyone completed what was assigned to them in a timely manner. The group worked well together and completed an equal amount of work.