Mitchell Hebert, Derek Anton

Outline

Essential Global Variables

File diskFile,RandomAccessFile disk, inputcommands, buffer
diskFile is used to read in the filesystem file that is saved on disk, and store it in disk
inputcommands take the list of commands passed in by the command line
buffer is used for the read() and write() functons, where it exists as a dummy buffer to read and write too before actually mutating the filesystem on disk.

Major Functions

CreateFS class

This will initialize our file system, formating it properly to be 128kb in size, with a 1st kb dedicated to our superblock. This super block is comprised of the following:

first 128 bytes holds the free block list
next 16 bytes hold the inodes. Each inode contains...

char[8] fileName
int intSizeOfFile
int[8] blockPointers
int ifUsed

int create(char[] name, int size)

create begins by reading in the free block list into a stack variable, and iterates through the list to check to see if there is enough space to create the newly requested file. We then seek further into the disk to see if each node is in use or not. If there is space and it is not in use, we start writting to each specific item in the free block (ie the new freeblock list, the new name, the new insizeoffile, blockpointers etc..) returns 1 if there was space, else returns zero.

int delete(char[] name)

we begin by reading in the name of the desired inode to delete, and then iterate through the disk looking for used blocks. If it is used, we read in the name of that block, and compare it to the requested name. If it is a match, we know to delete this, else keep searching.

int ls()

This was a simple function to implement, as we iterate through the disk we find the inodes that are used and print out to the console the important data

int read(char name[], int blockNum)

in the read function we take in the name of the target file to read, as well as the blocknumber. We then go through similar logic as the delete function to find the corect node, and check to see if the selected item has the same name as the requested file name to be read. If it matches, then we read this selected item into our global heap variable buffer (the dummy buffer) so we do not have to use up disk space in our file system.

write(char name[], int blockNum)

in the write function we take in the name of the target file to read, as well as the blocknumber. We iterate through the dummy buffer heap variable to search for the reqeusted file, if we have a match we take this file and check to see if we have space for it on disk. If there is space, we being writting to the free blocks avaliable to us and complete.