CSCI 6730 PROJECT 4
README

Xiaodong Jiang

April. 30th, 2017
Department of Statistics
University of Georgia

In this project, we are requested to implement a simple UNIX-like file system simulator in order to understand the hierarchical directory and inode structures, where we should be able to (1) browse the disk information, file and directory list, (2) create and delete files and directories, (3) read and write files.

1. Part I

   We are going to simulate a UNIX-like filesystem with small size files, and implement ‘df’, ‘create’, ‘stat’, ‘cat', 'read', 'write' and 'rm' functions. Please find more details about these implementations for ‘cat’, ‘read’, ‘write’ and ‘remove’.

Name	Note
cat	Based on the value of "inode.blockCount", we could find out how many blocks the small file (we are looking at) uses and read contents from those blocks.
read	We can find out that the output starts from "inode.directBlock[begin]" and "inode.directBlock[end]", based on the corresponding offset and size. Then we read contents from block "begin" to "end", as many as "size" strings defines before.
write	Here we can find out the block numbers and output contents. Then we write the "buf" (given by the user into the disk). Thus, the “buf” contents replace the original contents.
rm	We get the inode number and block numbers which contain the file contents. We first remove the inode number and block numbers from the "inodeMap" and "blockMap", and then update "superBlock.freeInodeCount" and "superBlock.freeBlockCount", and well as updating entry number of current directory.

2. Part II

   Now we are going to implement hierarchical, tree-structured directory, and implement ‘ls’, ‘mkdir’, ‘rmdir’ and ‘chdir’ functions. Please find more details below.

Name	Note
mkdir	We take the new directory as an entry under the currect directory, which contains name and inode number. Now the new directory becomes a directory itself, its first entry is the current directory "." and 2nd entry is its parent directory "..".
rmdir	We only remove one single inode number and one single block number.
chdir	Attention, before we change the current directory, don't forget to save it. We simply directly load the target directory from the block because we already know the directory inode number.

3. Part III

In this section, I extend the file system to support large up to 70,656

Name	Note
create	The first 5119 byte save into 10 direct blocks, then the rest size of file store into indirect block. Here we define a c struct to store the block data references.
cat	We read all the data from both direct blocks and indirect blocks.
read	We should pay attention to that if the require contents both in direct block and indirect block
write	As same as read function above.
rm	We need to remove indirect blocks