Project #3: File System Simulator
Design Document

Jared Willard Eric Kuha

December 4, 2017

Kernel.link
Kernel.link() takes two paths as parameters. It locates the first parameter and acquires an instance of its inode and inode number. The second path is used to create a new directory entry linked to the same inode creating a hard link. In the process, the method increments the inode's **nlink** attribute before writing it to the filesystem.

Kernel.unlink
The Kernel.unlink() method takes a path name as a parameter. It locates the directory entry and node. It shifts all entries after it in the directory up one slot. If nlink is 0, it then deallocates all blocks pointed to by the inode and finally, it truncates the size of the target directory by one, cutting off the last file (which was already shifted up by one). The test program, called **rm.cc** is effectively identical in function to the basic bash command 'rm'.

Indirect Blocks
Indirect blocks are functionality added to the IndexNode class. It is really quite straightforward. When the file is too large for direct block allocation, a pointer to an array of integers is allocated with a further ten block numbers to allocate. Functionality related to indirect blocks is added in the constructor (where the block is initialized to an array of empty pointers), the getBlockAddress() method which knows to look for any block number higher than MAX_DIRECT_BLOCKS in the level one indirect block array, the setBlockAddress() which works much like its respective getter, and then the read() and write() methods which write all of the inode's information to a buffer which is then written to or read from the disk. It's a straighforward concept and its implementation is similarly straightforward.

fsck.cc
A file system checker is a critical piece of programming for several reasons. It aids in garbage collecting orphaned disk blocks, directory entries, and inodes (though ours doesn't bother to reclaim inodes). It ensures that all inodes have the correct nlink value, accurately describing the number of directory entries that point to it. Next, it verifies that all blocks mentioned in inodes are marked as allocated blocks, and all other blocks are also marked free.