CSCI 5103 – Project 3
Rohit Sindhu [sindh010] || Aravind Alagiri Ramkumar [alagi005] || Aparna Mahadevan [mahad028]

Design Document

Linking files

- Get the inode number of the source file using Stat.
- Open the target directory of the file to be linked using Kernel::open.
- Move to the last line in the directory using Kernel::readDir and then add the directory entry for this new file with the existing file's inode number using Kernel::write.
- Read the inode of the file (using filesystem's readIndexNode) and increment the nlinks count and again write back the inode. (using filesystem's writeIndexNode)

Unlink files:

- Open the directory of the file to be unlinked using Kernel::open and obtain the file descriptor using the open files in the filesystem.
- Read the directory entries till the file entry.
- Read the next entry and move the offset back by one directory entry.
- Overwrite the entry read last at this current offset to delete the actual file entry to be deleted.
- Move the offset again to the correct position.
- Repeat the last 2 steps until all the entries are shifted back by one offset.
- Write the last entry as empty entry and decrement file size by one directory entry.
- Decrement the nlink count by 1.
- If the nlinks counts becomes 0, free all the direct/indirect blocks using file systems free blocks and setting the relative block address in inode to NOT_A_BLOCK.

Indirect blocks:

- In FileDescriptor::WriteBlock, if the relative block address is more than number of direct blocks, we check if the indirect block is allocated or not.
- If the indirect block is not allocated, we allocate the block and resume. While allocating the block we set all the entries of indirect block as NOT_A_BLOCK.
- We changed IndexNode's setBlockAddress and getBlockAddress to transparently set and get the relative block number address from direct or indirect block

  Changes in getBlockAddress:

  If the relative block address has to be given from indirect block, we read the indirect block using filesystem read. We then retrieve the address of that block number from the read indirect block and return that number.

  Changes in setBlockAddress:

  If the relative block address has to be given from indirect block, we read the indirect block using filesystem read. We then write the address of that block number into the indirect block and write back the indirect block into the disk.

FSCK:

- Perform breadth first search starting from the root directory by maintaining the index numbers in a queue.
- Check nlinks count of directory by verifying the count of sub-directory present in the directory is equal to nlinks.
- Stored all file inodes in a map to maintain the count of the occurrences of those inodes.
- Stored all the allocated blocks in a map while performing the BFS.

  Checking file nlinks:

  After the whole file system is read once, the inodes map is iterated to check whether the recorded count is equal to the nlinks of the inode.

  Checking block allocations:

  Iterate over all the data blocks in system: -
  Case 1: If the bit is set, and not present in allocated blocks map, print error.
  Case 2: If the bit is not set and present in allocated blocks map, print error.

FYI: We have handled a special case for checking the nlinks count of root directory as in mkfs it is set as 3.
There is a function named induceErrors which can be used to test fsck.
We have also created test.sh and a wrFile.txt to test all functionalities.