Lab 08: FileSystems

Design:

My design uses a hash function that I wrote myself. It is a part of the patch that I submitted. The design principle is basic in all ways. Every file is divided into blocks of size 4kB(4096 bytes). I used the getHash() function to get the hash of every block of data. The hash function depends on the data, so every block with same data has the same hash values. Every file in the root directory has the hash value separated by commas. The filename is the filename of the original file. The contents are stored in /tmp/blockstore/ with filename as the hashvalue.

Implementation:

When a file is read, first the file is looked up in the root directory. The relevant file in the root directory contains the hashed values of the blocks, that are stored as files in /tmp/blockstore/. So these are read and the read data is shown. So block deduplication is played in this way. When a file is to be written, the file is first divided into blocks of size 4kB each. The hash values are then computed and written into the file, with the intended filename, separated by commas. Then the corresponding blocks are checked for deduplication(if a file with the same hash value as filename exists or not) in /tmp/blockstore/ and if a copy is found, this data is discarded because it will be repeated if we store it. If a copy of the block is not found, then a file with filename as the hash value is created and stored in the same directory. Deduplication is implemented in this way.

Correctness:

I created 3 files. A 4 kB, 8 kB and a 12 kB file. I copied them into the mount directory. When I looked up theie contents in the root directory, the files displayed the hash values separated by commas. Then I went into the

/tmp/blockstore/ folder and looked up the files with the filenames, the contents were intact. I performed the read and write operations and got the desired results.