Kunal Jangam ksj48

Simple_test: 1037399ms
Test_case: 25724ms

Notes: I changed the size of struct dirent to 256 in tfs.h by changing name to name[250].

Helper functions:
Readi and writei used % and division to get the correct block the inode was located in as well as the correct position within the block the inode was located in.

Get_avail_ino and get_avail_blkno searched the bitmap for an unset bit, set it, and returned its position.

dir_find read the inode from the disk, updated the access time, then looked through all the blocks the inode pointed to for the dirent with the same name as fname.

dir_add looked for the dirent with the same name as fname, if it wasn't found, then it looked through the blocks of the inode for a spot to add the dirent. If no spot was found and another data block could be allocated, another data block was allocated and the dirent was added. The inode was updated and the m_time was also updated for the inode.

dir_remove looked for the dirent with the same name as fname, if found, said its validity to 0 and updated the inode along with its m_time.

get_node_by _path seperated the path by / and called find on each part, stored the inode and used it to find the next part.

Tfs functions
Tfs_mkfs initialized and opened the disfile, set up the super block, inode bitmap, datablock bitmap, and root inode and wrote them to memory.

Tfs_init called tfs_mkfs if the system was not initialized, otherwise opened the diskfile and read the super block from the disk into memory.

Tfs_destroy closes the file system.

Tfs_getattr calls get_node_by_path and fills in stbuf. If the inode represents a directory, st_mode was set to S_IFDIR | 0755. If it represents a file, st_mode was set to S_IFREG | 0666.

Tfs_opendir called get_node_by_path to make sure the directory exists.

Tfs_readdir called get_node_by_path and filled in buffer with the data from the inodes blocks using the filler function.

Tfs_mkdir seperated the basename and directory name of the path using dirname() and basename(). It then called get_node_by_path to get the inode of the directory. It called get_avail_ino to get an inode for the new directory. The new directory was added using dir_add and a new inode was created and written to disk.

Tfs_rmdir seperated the basename and directory name of the file using dirname() and basename(). It then called get_node_by_path twice to get the inode of the parent directory and the child directory. The block and inode bitmaps are updated and dir_remove is called.

Tfs_create was implemented the same way as tfs_mkdir but the new inode had type set to 0 to specify it was a file not a directory.

Tfs_open was implemented the same way as tfs_opendir.

Tfs_read used % and division to find which block the offset started in, and copied data from that position to the buffer. If more blocks were needed, the next blocks were read and copied until size number of bytes had been read.

Tfs_write first checked if the inode had enough blocks to write the required amount. If not, the blocks were allocated. Then starting from offset data was copied from buffer into the blocks, and additional blocks were loaded if needed until size number of bytes had been written.

Tfs_unlink was implemented the same way as tfs_rmdir.