Brian Schillaci (netid: bcs115)
John Strauser (netid: jps313)
05/05/19

CS416 Project 4: Tiny File System using FUSE Library Report

Descriptions of implementation:

Get_avail_ino

- Use bio_read to get the inode bitmap from the disk. Loop through until an unset bit is found. If the first unset bit is greater than the maximum number of inodes, return -1 for error. Otherwise, set the bit, write back to the disk, and return the inode number.

Get_avail_blkno

- Same as get_avail_ino except use the data block bitmap instead of the inode bitmap.

Readi

- Return an error if the inode number passed is greater than the maximum number of inodes. Use getInodeBlockNum to get the block that the inode exists in. Use getInodeBlockIndex to get the index that the inode is stored in from the previous block. Read the block containing the inode to buf. Copy the inode to arg inod. Return.

Writei

- Return an error if the inode number passed is greater than the maximum number of inodes. Use getInodeBlockNum to get the block that the inode exists in. Use getInodeBlockIndex to get the index that the inode is stored in from the previous block. Read the block that will contain the inode to buf. Copy the arg inod to the location of the inode in the block. Write the block back to disk. Return.

Dir_find

- Use readi to get the inode passed by the arg ino. If the inode is not a dir, return -1 for error. Loop through the direct pointers of the inode. For each valid pointer, Check if the block contains the dirent being searched for. If it does and it is valid, copy the dirent to the arg dirr and return. If not found, return -1.

Dir_add

- Use dir_find to check if a dir already exists with the same name. Scan through the pointers in the inode. If an empty spot exists, add the dirent at that location. If no data blocks are available, return -1. If the dirent was added successfully, return 0.

Dir_remove

- Search the dir_inode arg's data blocks for the fname arg. If found, remove the dirent and unset the bitmap. Decrement the number of links for the parent dir. Return 0. If the fname doesn't exist in the parent directory, return -1.

Get_node_by_path

- Use string tokenizer to separate the path arg by "/". If the token is NULL use readi to get the inode from the arg ino and return 0. Use dir_find to get the dirent into target. If dir_find fails, return -1. Otherwise, recursively call get_node_by_path on the remainder of the path.

Tfs_init

- Use dev_open to attempt to open the diskFile. If successful, initialize the super block from the diskFile. If unsuccessful, calls tfs_mkfs to initialize the file system.

Tfs_destroy

- Just free the superblock and call dev_close.

Tfs_getattr

- Use get_node_by_path to get the inode provided by the path arg. Depending on the type of the inode provide the args of the node to stbuf. Provide the final args to stbuf and return.

Tfs_opendir

- Use get_node_by_path to get the inode from the path arg. Return the result.

Tfs_readdir

- Use get_node_by_path to get the inode from the path arg. If the node cannot be found or it is not a directory, return -1 for error. Read the directories from the pointers in the inode. Copy each directory to filler.

Tfs_mkdir

- Separate the path names. Call get_node_by_path on the parent path. Use get_avail_ino to get the next inode number. Use dir_add to create the dirent in the parent directory. Create the inode for the new directory. Set the bitmap. Create the dirents for "." and "..". Write to the diskfile. Use writei to write the new inode).

Tfs_rmdir

- Separate the path names. Call get_node_by_path on the parent path. If the path provided is not a directory or has sub directories or files, return -1 for error. Clear the data block bitmap for the target directory. Clear the inode bitmap for the target directory. Clear data blocks associated to target directory. Clear inode block for target directory. Call dir_remove for parent directory.

Tfs_create

- Separate the path names. Call get_node_by_path on the parent path. Use get_avail_ino to get the next inode number. Use dir_add to create the dirent in the parent directory. Create new inode for target file. Ensure pointers are 0. Use writei to write the new inode.

Tfs_open

- Same as tfs_opendir.

Tfs_read

- Use get_node_by_path to get inode from path arg. Determine if there is enough space after offset to write size bytes. If not, return -1 for error. Scan through each block starting at offset and write to buffer each time a block is found. Repeat until enough bytes have been read.

Tfs_write

- Similar to tfs_read, except it creates blocks if blocks are not already in use.

Tfs_unlink

- Scan through each data block from inode provided by path arg. Zero out the data blocks associated and unset the bitmaps as needed. Call dir_remove for parent node to remove target file.

Total number of blocks used when running sample benchmark:

Running simple_test.c used around 110 blocks. This does not include blocks used by superblock and bitmaps that are made before sample_test.c runs.

Benchmark performance (time to run):

```
real	0m0.036s
user	0m0.001s
sys	0m0.003s
```

Detail for running code:

Just make and use ./tfs