Partners names and netids
Proj 4
Nhc29 - Nicholas Chen
Pcn32 - Paul Niziolek

Details on the total number of blocks used when running the sample benchmark, the time to run the benchmark, and briefly explain how you implemented the code

For the sample benchmark, we added our own testing file called testing.py (local testing since iLab goes down sometimes). This also allowed us to time before and after running the code. Here, our testing file would delete all remnants of the file, recompile everything, and then run the tests. This would make it a lot easier on ourselves, the developers, since we did not need to remember to clear the mount directory, the disk file, and remake our changes.

For the simple test, our python script measured that it would finish it in ~0.02489s
For the test_case test, our python script measured that it would finish it in ~0.02861s

As for our implementation, we did it very straightforwardly.

get_avail_ino: This function retrieves the bitmap's available inode number. In order to locate an open slot, it reads the inode bitmap from the disk and navigates through it. It modifies the bitmap and writes it back to the disk if a slot is discovered.

get_avail_blkno: In a similar manner, this one uses the bitmap to retrieve the number of accessible data blocks. It locates a free slot, reads the data block bitmap from the disk, modifies it, and then writes it back to the disk.

readi: An inode is read from the disk using this function. It reads the block from the disk into a temporary buffer, determines the block number and offset where the inode is located, and then copies the desired inode into the output parameter.

writei: The disk is used by this method to write an inode. It reads the block from the disk into a temporary buffer, copies the updated inode data to the proper spot in the buffer, determines the block number and offset where the inode is located, and finally writes the block back to the disk.

dir_find: This function searches a directory for an entry with a specified name. It retrieves the directory's inode, reads every directory block, and examines every directory item. The entry is copied into the output parameter if one with the specified name is found.

dir_add: Adds a directory entry to a directory using this function. Every directory block is read, and every directory entry is examined. The function gives an error if there already exists an entry with the same name. If not, it locates a slot that is free, modifies the directory entry, and then writes the block back to the disk.

dir_remove: A directory entry can be removed from a directory using this function. Each directory block is read, the item with the specified name is located, it is invalidated, and the block is written back to the disk.

get_node_by_path: Using its path, this function retrieves an inode. After tokenizing the path, it locates each token's directory entry and reads the associated inode.

rufs_mkfs: The file system is initialized via the rufs_mkfs function. The diskfile is initialized, the superblock information is written, the bitmaps for the data blocks and inodes are initialized, the root directory's bitmap is updated, and the inode is updated. Here, we separated the initialization into helper functions for simplicity sake

rufs_init: This function initializes the file system. It opens the disk file and if it fails, it calls rufs_mkfs to create a new file system. It then allocates memory for the superblock, inode bitmap, and data block bitmap, and reads these structures from the disk.

rufs_destroy: This function cleans up the file system. It frees the memory allocated for the superblock, inode bitmap, and data block bitmap, and closes the disk file.

rufs_getattr: This function gets the attributes of a file. It reads the inode of the file and copies its stat structure to the output parameter.

rufs_opendir: This function opens a directory. It reads the inode of the directory and checks if it’s valid.

rufs_readdir: This function reads the entries of a directory. It reads the inode of the directory, reads each directory block, and checks each directory entry. If an entry is valid, it calls the filler function to add the entry to the output buffer.

rufs_mkdir: This function creates a new directory. It gets the inode of the parent directory, finds an available inode number, adds a directory entry to the parent directory, initializes a new inode for the new directory, and writes the new inode to the disk.

rufs_rmdir: This function removes a directory. It gets the inode of the directory, invalidates all its data blocks, invalidates the inode itself, gets the inode of the parent directory, and removes the directory entry from the parent directory.
rufs_releasedir: This function is a placeholder and doesn’t need to be filled for this project.

rufs_create: This function creates a new file. It gets the inode of the parent directory, finds an available inode number, adds a directory entry to the parent directory, initializes a new inode for the new file, and writes the new inode to the disk.

rufs_open: This function opens a file. It reads the inode of the file and checks if it’s valid.

rufs_read: This function reads data from a file. It reads the inode of the file, calculates the block index and offset for each byte to be read, reads the corresponding block from the disk, and copies the data to the output buffer.

rufs_write: This function writes data to a file. It reads the inode of the file, calculates the block index and offset for each byte to be written, reads the corresponding block from the disk, copies the data from the input buffer, and writes the block back to the disk.

Any additional steps that we should follow to compile your code

Just call make on the root directory. Then begin mounting

Any difficulties and issues you faced when completing the project

Not many difficulties and issues besides the fact that we needed the extra 2 days to submit our project.

Collaboration and References: State clearly all people and external resources (including on the Internet)

that you consulted. What was the nature of your collaboration or usage of these resources

ChatGPT to ask about proper code quality and leading us hints to fill functions. Also helped describe steps to completing some functions which was convenient