Design Explanation

MRFS

Subham Rajgaria 15CS10061

Sunaina Singh Rathod 15CS30023

Group No - 35

Introduction

Structures used

Superblock

The superblock was used as a structure and it included all the details of the file system and the bitmap of inode was taken of size 2 as we assumed that **maximum inodes is 32** and data blocks bitmap was taken of size **MAX_DATA_BLOCKS/32 +1**. MAX_DATA_BLOCKS was taken such that total file system was of size **1GB**. We padded the superblock so as to give superblock a size in multiples of **256**. It also included the **current working directory** parameter.

Inode

The inode structure contained details about owner access permission, etc. We added an attribute called file size in the inode so as to keep a count of the files in case of a directory. Each inode was given a size of **256** bytes though it was of a smaller size than that. **The first inode was reserved for root. Each directory was assumed to contain a maximum of 64 files.**

File Table

The file table was defined as a map **<int, file_desc>** . The file_desc was a struct which contained the inode no, the mode of the file and the read/write offset till now.

Global Variables

The file system was declared as a global char * pointer and the file_system_size and the max_fd allotted to any open file was also kept global.

Helper Function:

- long FdGetFileSize(int fd) :
  This function is used to get the file's size from its file descriptor fd.
- void setter(int i, int a[ ]):
  This function is used to set 'i' in the bitmap 'a'.
- void clr(int i, int a[]) :
  This function is used to clear 'i' from the bitmap 'a'.
- int test(int i, int a[]):
  This function is used to test whether 'i' is set in the bitmap 'a', it returns 1 of it is set and 0 if it is not set.
- int get_next_empty_block():
  This function returns the next empty data block number starting from the first data block.it uses the superblock's db_bitmap to test for the empty data block
- int get_next_empty_inode():
  This function returns the next empty inode block number starting from the first data block.it uses the superblock's inode_bitmap to test for the empty data block.
- int get_file_inode(superblock *temp, char *filename)
  This function returns the inode number of the file specified the filename if it exists else it return -1.
- int remove_file_db(superblock * temp, inode* curr_inode):
  This function is used by the remove_myfs function to remove the data blocks of the file whose inode is specified as curr_inode, it returns 1 is it is successful else it returns -1.

Function Design

Create_myfs

File_system was malloced and root directory was created and . file was created in the root directory.

Copy_pc2myfs

The filename to be copied was first alloted a inode if available and the it was updated in the parent directory's list.Then the data was transferred from the PC to MRFS's data blocks and their metadata was updated in the inode's fields.

Rm_myfs

File_inode was obtained and all the datablocks were cleared from the bitmap and the datablocks were made bzero. Also the inode was cleared from the bitmap in the superblock. The file size of the file was decreased from all the directories above it in the hierarchy.

Showfile_myfs

Showfile_myfs was modified to take another parameter as it helped in copy_myfs2pc where instead of cout we could directly write to the file.

Mkdir_myfs

Mkdir_myfs is used to make the directory in the current directory, the directory is initialised with a free inode if available and the corresponding data block is also initialised with the '.' and '..' entries and their corresponding inode numbers.

Rmdir_myfs

Here we first get the inode of the directory and change the current working directory to this and then delete all the files from that directory using file count. Once we delete all the files from there we go back to parent directory and clear the inode of the directory to be deleted.

Open_myfs

Open_myfs is used to open the given file in one of the following access modes either 'w'(write) or 'r' (read) , the access permissions are to be satisfied to successfully generate the File descriptor. The corresponding entry is made into the file desc table, which contains all the FD of currently open files along with there inode numbers and the pointer to the next position in the file.

Read_myfs

Read_myfs function is used to read nbytes of data . Based on the read offset and bytes to be read we first get the start block and end block for the file. Now we do a conditions check and find where do we have to read from : in the direct blocks, indirect block or doubly indirect block . Then depending on that we find the block offset and read 256 bytes at a time.

Write_myfs

Write_myfs function is used to write nbytes of data . Based on the write offset and bytes to be written we first get the start block and end block for the file. Now we do a condition check and find where do we have to write : in the direct blocks, indirect block or doubly indirect block and from that place we copy nbytes of data sequentially.