Assignment 4 :Implement a Memory Resident Unix-like File System Documentation

Group Members :-

1. Aayush Gupta (15CS10001)
2. Vivek Gupta (15CS10053)

In addition to the design details mentioned in the assignment statement for the file system, following are the other details implemented:

1) The file system consists of three regions: (a) super block, (b) inode list, (c) data blocks.

2) The super block contains information about the file system :- total size, maximum number of inodes, actual number of inodes being used, maximum number of disk blocks, actual number of disk blocks being used, and bitmaps of disk blocks and Inodes. Also, inode number of root directory.

3) The number of inodes is pre-decided according to the macro INODE (set to 1024).

4) Number of data blocks is calculated dynamically depending on the MRFS size specified by the user :-
   (MEGA*size - (5*sizeof(int) + INODE*(sizeof(char)+sizeof(inode)))) / (sizeof(char)+BLOCK_SZ)

Macro variables :-

- MEGA = (1024*1024)
- INODE = 1024
- BLOCK_SZ = 256
- MAX_OPEN_FILES = 1000

Global Variables :-

- void * myfs :- Pointer to file system
- void * myblocks :- Pointer to first data block
- sem_t * sem,*sem2 :- Semaphore variables
- int shmid, shmid2 :- shmids of shared memory (fs & file table)
- int * iptr :- Integer pointer to super-block
- int cwd :- Inode number of current working directory
- int open_files :- number of currently open files

- char *inode_bmap :- Bitmap of inodes
- char *block_bmap :- Bitmap of data blocks
- inode * myinode :- Pointer to first inode
- file **myfiles :- Global file table

Structures :-

1) **inode**: To represent an inode

Fields :-

1. short file_type
2. short access_permission
3. int file_size
4. time_t last_modified
5. time_t last_accessed
6. int block_ptr[10] : First 8 are direct pointers; 9th one is singly indirect and 10th one is doubly indirect.

2) **file**: To store information in the global open file table

Fields :-

1. int inode_no
2. int offset

3) **block**: Structure to store information about a data-block

Fields :-

1. int no : data-block number
2. int nbytes : remaining bytes in block
3. void *empty : pointer to the first empty byte

Functions

1) **short bin2short(char *bin)**

Converts 9 bit char access permission to short type to reduce memory usage and returns it.

2) **void update_super()**

Updates number of data blocks and inodes used according to their bitmaps.

3) **int create_file_entry(int inode_no, int offset)**

Inserts *inode_no* and *offset* (0 for newly opened file) of file in the global table of open files. *Returns the file descriptor* of the opened file.

4) **int get_freeInode()**

Scans inode bitmap and returns index of first unused inode and marks it used.

5) **int get_freeDB()**

Scans data block bitmap and returns index of first unused data block. Also, marks it used.

6) **void newInode(int inode_no,short file_type,short file_size)**

Initialises newly created file or folder inode

7) **block newDB(int inode_no)**

Returns a new data block where data can be written and attaches it to the inode with index *inode_no*.

8) **void add_file2dir(int folder_no, char *fname,int file_no)**

Inserts file entry with inode number *file_no* and name fname to the directory with inode number *folder_no.*

9) **int clear_myfs()**

Detaches shared memory segment for file system and global file table. Also unlinks and closes the semaphores.

10) **int getFileInode(char* fname,int rem_file)**

Returns inode number of file with name *fname* inside the current working directory. If flag *rem_file* is set to 1 then the file entry is removed from the current working directory.