

FATBoys Floppy File System

Description

In this project we designed and implemented a floppy file system library for Linux operating system in C language.

The FATBoys floppy file system looks like the below figure:

Super Block	Bitmap		Fat Array						Directory Entry Table		File blocks	...					
0	1	2	3	4	5	6	7	8	9	10	11						1439

Super Block: Contains struct `ffs_sb_info ffs_sb`. It is 41B and resides on first block of floppy.

```
struct ffs_sb_info
{
    char* fileName;
    int noOfDirectoryEntries;
    int fatLength;
    int noOfFats;
    int bufSize;
        int noOfFiles;
        int remainingBytes;
        int remainingBlocks;
    int s_blocks_count;
    int s_byte_count;
    int s_block_size;
};
```

Bitmap: It is a character array representing the blocks of the floppy. It is 1440B and resides on 1st and 2nd blocks of the floppy. Each empty block is represented by 'e' and used blocks by 'u'.

Fat Array: Fat array is a linked list implemented using an integer array. It links the blocks of a file to each other since the blocks of a file may not reside contagiously. It is 5760B and resides on 6 blocks (3-4-5-6-7-8) of the floppy.

Directory Entry Table: It is an array of struct `dirEntry`. Contains such information as the name,size of the file and its start address on the floppy. Maximum number of files allowed is 50 and it is 1850B and resides on two blocks (9-10).

```
struct dirEntry
{
    char* name;
    int noOfUsers;
    int currentStatus;
```

```
int size;  
int firstClusterAddress;  
int empty;  
int permission;  
int isOpen;  
int lastClusterAddress;  
int bytesInLastBlock;  
};
```

File Blocks: Files are created and written by 1024B blocks on the floppy. They may consist of multiple blocks which are scattered on the floppy. They are fetched by first locating them on the directory entry table, finding their first cluster address, and then fetching the next blocks by following the links on the FAT.

Some of the Global Variables

struct ffs_sb_info: Contains information about the filesystem. Name of file system, number of directory entries, number of files, remaining bytes and blocks in the floppy, total number of bytes and blocks of the floppy, size of a block are some of the fields.

struct dirEntry: Contains information about name of the file, number of users (allows sharing of file for reading), current status of file(read, write), size of file, first and last cluster addresses, number of bytes in the last block are some of the fields.

struct fileOffset: Contains the byte offset. If a user wants to append to the file, write operation is performed after the byte offset in the file.

int fatArray[NUMBER_OF_BLOCKS]: We keep a file allocation table array to mirror the floppy disk's blocks.

char bitmap[NUMBER_OF_BLOCKS]: This bitmap array helps us find the next available free blocks on the floppy.

struct fileOffset offsetArray[MAX_NO_OF_FILES*MAX_NO_OF_USERS]: File offset keeps track of where each user till the file is closed by that user.

Functions of the library

int ffs_dumpfloppy(char *filename): This function reads the floppy and outputs the content of each block into a text file named "filename". Returns 0 on success, -1 otherwise and sets corresponding error indicators.

int ffs_makezero(void): Resets all bytes of the floppy to zero. Returns 0 on success, -1 otherwise and sets corresponding error indicators.

int ffs_format(char *devname): Install the FATBoys file system on the floppy on the special file "devname" which corresponds to the floppy disk. It counts the number of bytes on the floppy and calculates the number of blocks. It initializes "ffs_sb", bitmap, directory

entry table, and the FAT array accordingly. It flushes this information on the floppy by calling "ffs_sync()". Returns 0 on success, -1 otherwise and sets corresponding error indicators.

int ffs_mount(char *devname): Special file corresponding to floppy is opened by this function and is not closed till "ffs_umount()". Creates the necessary data structures(super block, bitmap, FAT, and directory entry table) by reading them from the floppy. Returns 0 on success, -1 otherwise and sets corresponding error indicators.

int ffs_umount(): Flushes all the information about the FATBoys file system to the floppy by calling "ffs_sync()" and closes the special file corresponding to the floppy disk. Returns 0 on success, -1 otherwise and sets corresponding error indicators.

void ffs_sync(void): Flushes the super block, bitmap, FAT, and directory entry table to the floppy.

int ffs_file_create(char *filename): Creates a file named "filename" on the floppy. It first locates the first empty directory entry and writes the ... It then finds the first empty block by iterating the bitmap array. This is the first cluster address of the file. The file has size=0 at creation. The corresponding blocks are then set in the FAT and bitmap. Returns 0 on success, -1 otherwise and sets corresponding error indicators.

int ffs_file_delete(char *filename): Removes the file named "filename" from the floppy disk. It first checks if the file exists by iterating the directory entry table. If the file is found, its blocks on the floppy are marked with zero, the bitmap and FAT are updated. All the information is flushed to the floppy. Returns 0 on success, -1 otherwise and sets corresponding error indicators.

int ffs_file_open(char *filename, int flags): The function takes, as arguments, a filename to open and an integer flag which indicates that either the file will be opened for reading or writing. If there exists such file to open, function checks whether the file is already opened. If that is the case, it is checked that, if the requested permission can be granted. A file which is already opened for reading cannot be opened again for writing but reading. A file which is already opened for writing cannot be opened again for any kind of transaction. The function returns an integer file descriptor to access the file for read, write and close operations. In the case of errors like having non-existing filename as argument, the function returns -1 and sets corresponding error indicators.

int ffs_file_close(int fd): This function takes an integer file descriptor as an argument and closes the corresponding file which means preventing the access of the file for read, write and close operations by using the given file descriptor. If there are multiple users reading the file, the number of users is decremented by 1 each time this function is called, and when the number of users equals 0 then "isOpen" field in the directory entry table corresponding to the file is set to "false". If the file is already closed, the function returns -1 and sets corresponding error indicators, otherwise it returns 0.

int ffs_file_read(int fd, void *buf, int size): The function takes 3 arguments; an integer file descriptor to access the corresponding file, a void pointer to a location in the memory to

copy the content file, an integer size, which indicates how many bytes are requested to read from the file. If the file is already read with the same file descriptor, when another read operation is requested, the file will be read from the point that the previous read operation ended. The size of the content of the file which is read is returned by the function if no error occurs. Otherwise it returns -1 and sets corresponding error indicators.

int ffs_file_write(int fd, void *buf, int size): As argument, the function takes an file descriptor to access the file, a void pointer to a location in the memory to write into the file and the third argument indicates the length of the memory part which is going to be written into the file. If the file does not exist or if there is no enough space in the disk, the function returns -1 and sets the corresponding error indicators. Otherwise it returns the length of the memory portion which is written.

void ffs_dir_list(): The ffs_dir_list() function prints the name and the size of the files which exist in the disk.

void ffs_printinfo(): The ffs_printinfo() function prints the following information about the disk.

- The file system name which is used in the disk
- Size of a block
- The number bytes written in the disk
- The number of blocks in the disk
- The length of the file allocation table

void ffs_printerror(): The ffs_printerror() function prints encountered errors, which are set in the other functions.