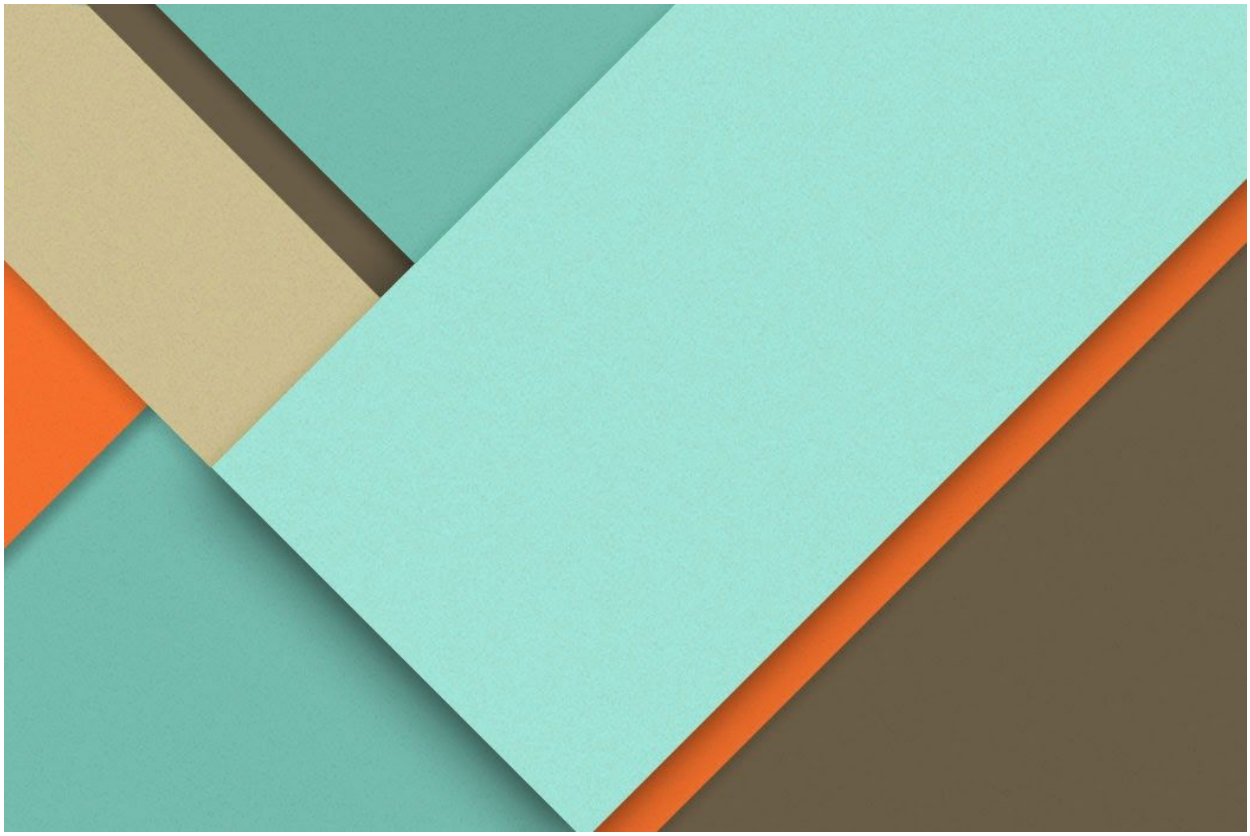


Operating Systems
Midterm Report

Muhammed Okumuş
151044017

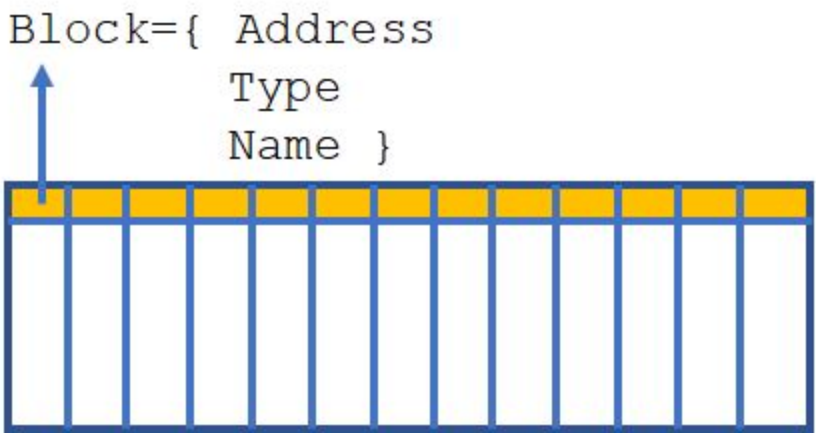
Overview

This report explains a simplified file system design. Parts of this design are inspired from the ext4 journaling file system.

Block Structure

The file system consists of a number of blocks. These blocks can hold the information about i-nodes, superblock, i-node bitmap and other types of data structures defined and explained later in the report.

Each single block has a small-ish structure reserved for addressing and defining the rest of the block. Being able to define blocks is essential since the system needs to know what it's retrieving and make meaning of the bytes.



Address is an integer representing the index of the block, ranging from 0 to N, where N is the number of blocks.

Type is ideally an enumerated integer. It can take values:

1. BLOCK_INODE
2. BLOCK_FILE_CONTENT
3. BLOCK_DIRECTORY
4. BLOCK_SUPER
5. BLOCK_INODE_MAP
6. BLOCK_FREE

While retrieving a block, the system will see the block type, then continue with a suitable structure appropriate to block type.

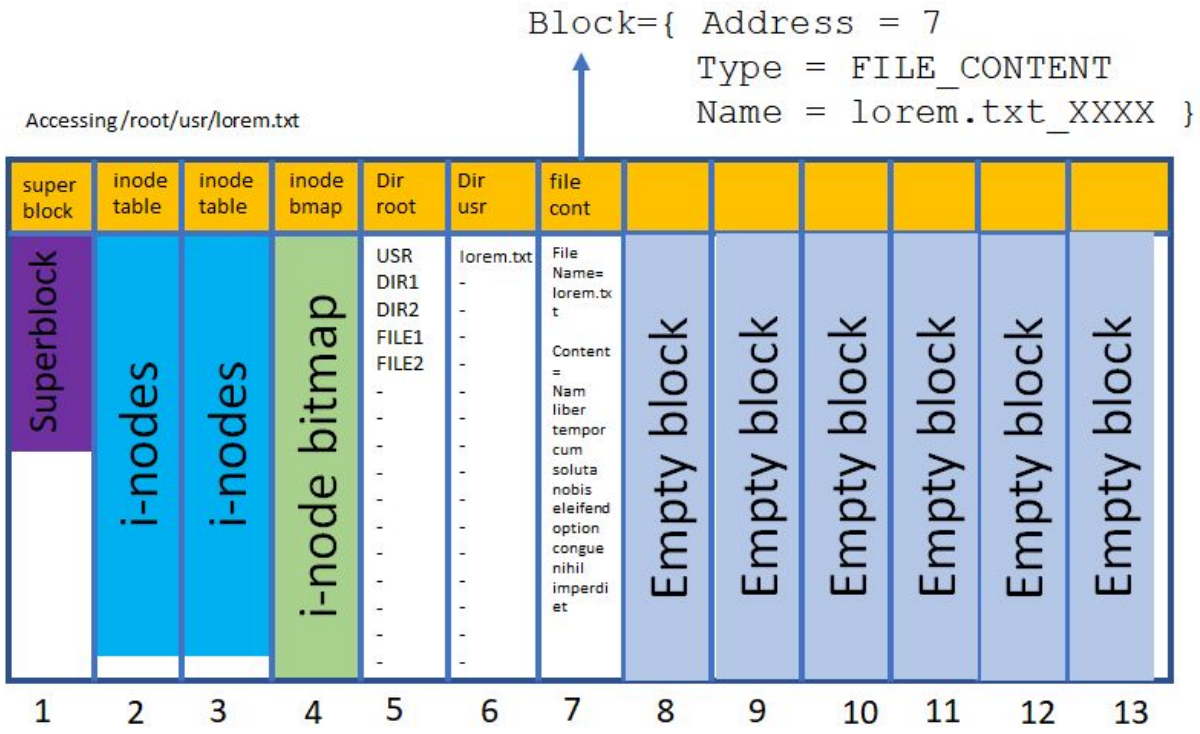
Name is kept for file/directories, it can also be useful for sanity checks.

Directory Structure

Directories hold unique strings in their blocks. Each directory occupies a single block and can have a limited number of sub-directory or file references indicated by the unique strings. The references are limited by the block size.

Walking a Directory

Each file or directory can be accessed by 1 loop through the file system blocks. This is possible because when each file or directory created the block is assigned a unique ID.



Upside of this approach is having very fast access times. Downside is the system uses more memory.

I-node Structure

A i-node holds:

1. Index
2. Direct block start address
3. Indirect block addresses
4. Type
5. Size
6. Time last modified

Index is used by the system to calculate inode location. For example the i-node of the root always has its index=0. To access Nth i-node, system makes the calculation:

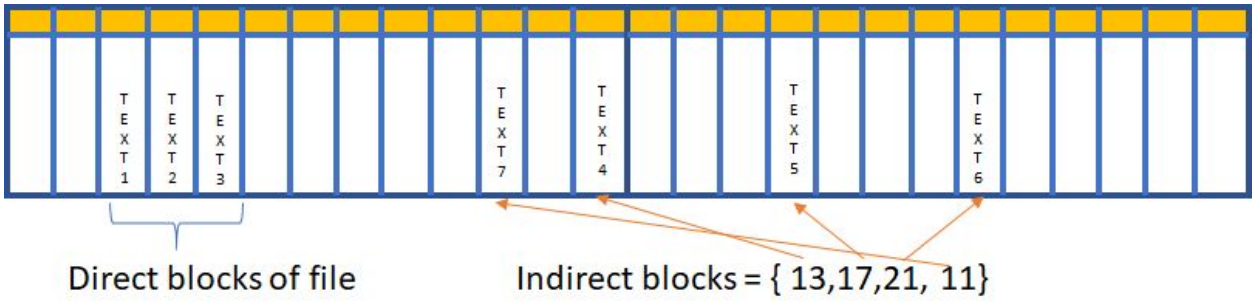
$$Address = Block\ Size * (1 + N/total\ inode\ blocks + N\%inodes\ per\ block)$$

Plus 1 comes from super block since it always occupies block 0.

Direct block start address points to a data block. This data block can be 3 types. First 2 are directory and file content blocks. Third one is a free type block. This may occur because of unexpected system shut downs.

Indirect block addresses are lists of pointers that point to data blocks. They also can point to 3 types of block like the direct address.

When a file is created, a predetermined number of blocks are reserved. If the file size exceeds the size of these predetermined blocks, indirect blocks are used. While the direct blocks consist of n to n+k back to back blocks, indirect blocks are not successive, they can be anywhere.



Type can be either file or directory.

Size is the number of blocks occupied by the file times block size for files. For directories it's equal to the block size.

Time last modified is set every time a block pointed to by the i-node is changed.

Superblock Structure

Super block is the most important part of the file system. Without it, the system can not tell what is in the file system or how to interpret it. When the file system is being retrieved from the disk, it will always check the first block of the file system for superblock and read the number of inode blocks, their locations, bitmap block etc. The provide such information the superblock holds:

1. State
2. # of blocks
3. # of free blocks
4. First block address
5. Block size
6. # I-nodes per block
7. # I-node blocks
8. # Free i-nodes
9. I-node size
10. Superblock size

State is used to check if an error occurred in the system, if the system was able to set it. This information can be useful to check block integrity if there is an error occurred.

# of blocks, # of free blocks, block size, # i-nodes per block, # i-node blocks and **i-node size** are all used for file system retrieval and write back address calculations.

Part 3 Functions

// Function Prototypes ======================

// Display/Debug Functions

void print_usage();

void print_superblock();

void print_inode(struct Inode i);

void print_inode_n(int i);

void print_bitmap();

// Getters for inodes/blocks information

char* get_name(int address);

char* get_block_type(struct Block b);

struct Inode* get_inode(int i);

int file_exists(char* path);

// Command Processing Functions ==============

int cmd_list(char* path);

int cmd_mkdir(char* path);

int cmd_rmdir(char* path);

int cmd_dumpe2fs();

int cmd_write(char* path, char* linuxFile);

int cmd_read(char* path, char* linuxFile);

int cmd_del(char* path);

int cmd_ln(char* path1, char* path2);

int cmd_lnsym(char* path1, char* path2);

int cmd_fsck();

// File system retrieval/booting

int read_filesystem();

// Bitmap access/modification

int read_bit(int b);

int set_bit(int b);

int clear_bit(int b);