Ceng334

Homework 3 Recitation

Fırat Ağış
firat@ceng.metu.edu.tr

May, 2024

Outline

1. Ext2 File system
   - Overview
   - Super Block
   - Block Group Descriptors
   - Bitmaps
   - Inodes
   - Directories
2. Assignment
   - Guarantees
   - Objectives
     - Bitmap Recovery
     - Pointer Recovery
     - User Conformation
3. Final Remarks

Overview

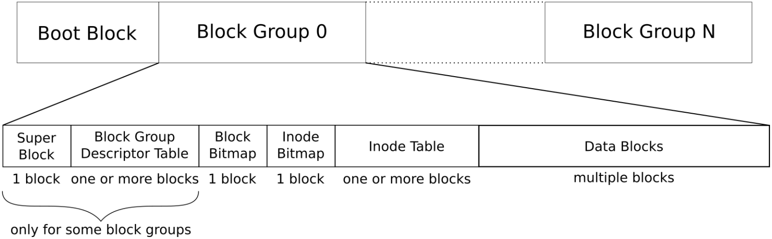
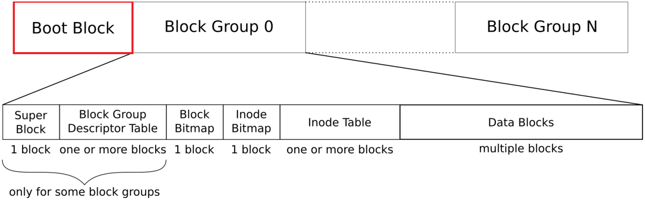


Figure: Overall ext2 Layout

Overview

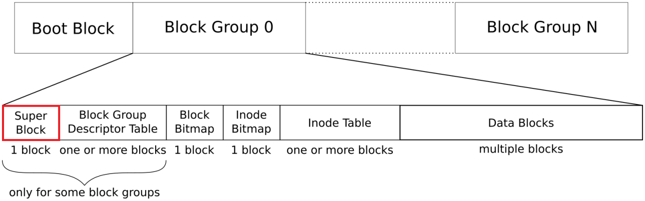
Boot Block



- Not relevant for our purposes.

Overview

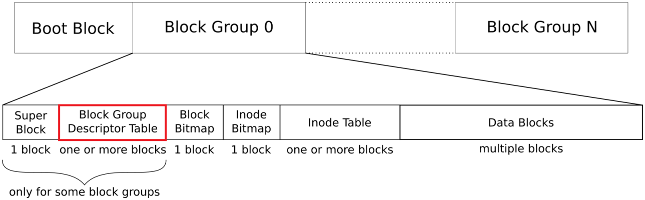
Super Block



- Properties of the whole file system (ie. block size).
- Generic properties of block groups (ie. block per block group)

Overview

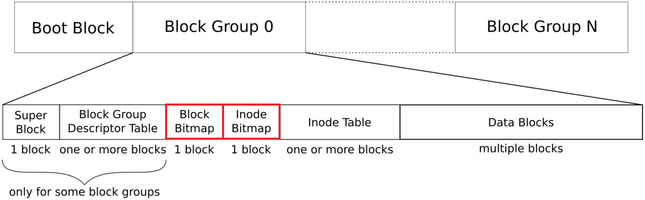
Block Group Descriptors



- Locations of important blocks in that block group.
- Properties specific to the current state of the block group.

Overview

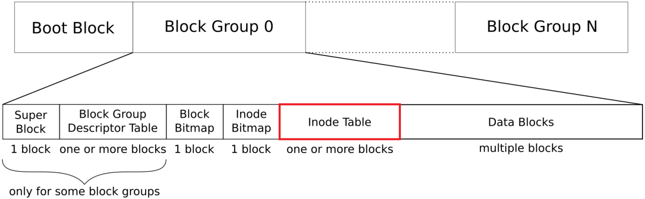
Bitmaps



- One for blocks, one for inodes
- Records whether a block or inode is in use or not.
- Specific to that block group.

Overview

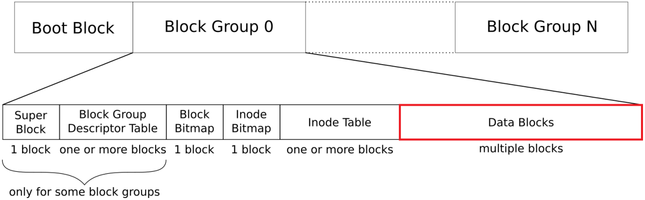
Inode Table



- Houses inodes located in the block group.
  - There are inodes for each file and directory.
  - Inodes records the properties of file/directories, as well as where their data is located within the filesystem

Overview

Data Blocks



- Blocks that contains data or pointers.

Super Block

This is not the complete picture, just the relevant fields.

```
struct ext2_super_block {
    uint32_t inode_count; /* Total number of inodes in the fs */
    uint32_t block_count; /* Total number of blocks in the fs */
    ...
    uint32_t free_block_count; /* Number of free blocks */
    uint32_t free_inode_count; /* Number of free inodes */
    uint32_t first_data_block; /* The first data block number */
    ...
```

- The super block always begins at byte 1024.
- `inode_count` and `block_count`: Total number of blocks and inodes in the file system.
- `free_block_count` and `free_inode_count`: How many of those inodes and blocks are not in use.
- `first_data_block`: Number of the first data block, depends on the block size.

Super Block

```
    uint32_t log_block_size; /* 2^(10 + this value) gives the block size */
    ...
    uint32_t blocks_per_group; /* Number of blocks for each block group */
    ...
    uint32_t inodes_per_group; /* Number of inodes for each block group */
    ...
    uint32_t first_inode; /* First non-reserved inode
            in the file system */
    uint16_t inode_size; /* Size of each inode */
    /* More, less relevant fields follow */
};
```

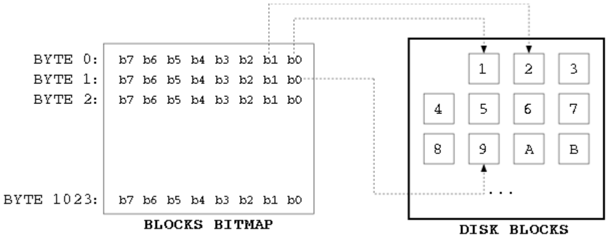
- `log_block_size`: Size of each block, given by the formula.
- `blocks_per_group` and `inodes_per_group`: How many blocks and inodes are within a block group. Last block group can have less than these.
- `first_inode`: First inode not reserved by the system.
- `inode_size`: Size of each inode.

Block Group Descriptors

```
struct ext2_block_group_descriptor {
    uint32_t block_bitmap; /* Block containing the block bitmap */
    uint32_t inode_bitmap; /* Block containing the inode bitmap */
    uint32_t inode_table; /* First block of the inode table */
    uint16_t free_block_count; /* Number of free blocks in the group */
    uint16_t free_inode_count; /* Number of free inodes in the group */
    uint16_t used_dirs_count; /* Number of directories in the group */
    uint16_t pad; /* Padding to 4 byte alignment */
    uint32_t reserved[3]; /* Unused, reserved 12 bytes */
};
```

- Always starts at the first unoccupied block after the super block.
- You should read the locations of bitmaps and inode tables from here.

Bitmaps


BYTE 0: b7 b6 b5 b4 b3 b2 b1 b0
BYTE 1: b7 b6 b5 b4 b3 b2 b1 b0
BYTE 2: b7 b6 b5 b4 b3 b2 b1 b0
BYTE 1023: b7 b6 b5 b4 b3 b2 b1 b0
BLOCKS BITMAP
1
2
3
4
5
6
7
8
9
A
B
...
DISK BLOCKS

- Bitmaps are always exactly 1 block and there are only 1 bitmap of each kind in a block group.
- A bit corresponding to an inode or a block is set to 1 if its in use, set to 0 if not.
- Each bitmap only contains information about the blocks/inodes in the block group it's in.

Inodes

```
struct ext2_inode {
    uint16_t mode; /* Contains filetype and permissions */
    uint16_t uid; /* Owning user id */
    uint32_t size; /* Least significant 32-bits of file size in rev. 1 */
    uint32_t access_time; /* Timestamps (in seconds since 1 Jan 1970) */
    uint32_t creation_time;
    uint32_t modification_time;
    uint32_t deletion_time; /* Zero for non-deleted inodes! */
    ...
};
```

- `mode`: File type (user file or directory for our purposes) and permissions for the file.
- `size`: Least significant 32-bits of file size.
- `access_time`, `creation_time`, and `modification_time`: Various timestamps.
- `deletion_time`: Also a timestamp. Zero if an inode is not deleted.

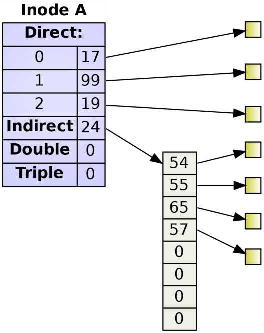
Inodes

```
    ...
    uint16_t gid; /* Owning group id */
    uint16_t link_count; /* Number of hard links */
    uint32_t block_count_512; /* Number of 512-byte blocks
alloc'd to file */
    uint32_t flags; /* Special flags */
    uint32_t reserved; /* 4 reserved bytes */
    ...
};
```

- `uid` and `gid`: User id and Group id of the user and the group that owns the file.
- `block_count_512`: Number of blocks allocated to the file if block size was 512-bytes. Can be use to calculate how many blocks are allocated to a file.

Inodes

```
    ...
    uint32_t direct_blocks[12]; /* Direct data blocks */
    uint32_t single_indirect; /* Single indirect block */
    uint32_t double_indirect; /* Double indirect block */
    uint32_t triple_indirect; /* Triple indirect block */
    /* Other, less relevant fields follow */
};
```



Directories

```
struct ext2_dir_entry {
    uint32_t inode; /* inode number of the file */
    uint16_t length; /* Record length, aligned on 4 bytes */
    uint8_t name_length; /* 255 is the maximum allowed name length */
    uint8_t file_type; /* Not used in rev. 0,
                    file type identifier in rev. 1 */
    char name[]; /* File name. This is called a
                    'flexible array member' in C. */
};
```

- Each directory contains the entries for "." and "..".
- If `inode` field is 0, the directory entry should be ignored.
- `length` field of the last directory entry in a block is set in a way that it fills the remaining space in the block.

Assignment

- You will be designing a recovery program for ext2 file system.
- You will be using C or C++ programming languages.
- Your assignment will be tested on inek machines.
- Your due date is 30 May 2024, Thursday, 23:59

Guarantees

- The data erasure is limited to bitmaps and pointer fields of inodes, everything else is completely intact.
- Contents of the inodes reserved by the system is also intact.
- Any unused data block is wiped clean, and contains no set bits.
- Any non-empty data block that contains user data (ie. not directory or pointer data) starts with the same 32-byte identifier that will be given at runtime.
- Pointers to data blocks that are empty are intact.
- First two parts of the homework are independent, but completing one might help you with the other one.

Objectives

Your goal is:

- Recover deleted bitmap bits.
- Recover deleted pointers on inodes.
- Print the structure of the file system.

Objectives

Bitmap Recovery

Bitmaps on the file system might be damaged.

- That damage can be partial.
- Not all bitmaps might not be damaged.
- Bitmap damage is limited to bits being set to 0.

Objectives

Pointer Recovery

Some pointers of inodes are set to 0 incorrectly, you need to fix it using (not limited to) the following informations:

- Data blocks in use that are not being pointed to.
- Inodes of "." and ".." directory entities.
- Sizes of files located in inodes.

Objectives

User Conformation

Finally you will print the recovered file system to the standard output in a tree structure, such as:

```
- root/
-- home/
--- user1/
---- file1.txt
---- file2.txt
--- user2/
---- file3.txt
-- etc/
--- config.txt
```

Final Remarks

- This presentation is to aid you, but it is not comprehensive, refer to the homework text for any specifics.
- For simplification, the outputs can be in any order.
- Including POSIX ext2/ext3/ext4 libraries (as well as kernel codes and their variations etc.) is **not** allowed.
- There will be no extension for this homework, use your time wisely.

Good Luck Everyone