

# CSE321 Project 2

## VSFSck: A Consistency Checker for Very Simple File System (VSFS)

In this project, you will **design** and **implement** a file system consistency checker, `vsfsck`, for a custom virtual file system (VSFS). Your tool will be responsible for verifying the **integrity** and **consistency** of essential file system structures, including:

- **Superblock**
- **Inodes**
- **Data blocks**
- **Inode and data bitmaps**

The checker will operate on a file system image (`vsfs.img`), identifying and reporting any inconsistencies found.

### Task Description

You will be provided with a corrupted file system image (`vsfs.img`) containing various errors. Your objectives are to:

1. **Analyze** the file system image using your `vsfsck` tool.
2. **Identify** all inconsistencies and structural issues.
3. **Fix** the detected errors to restore the file system's integrity.
4. Ensure that the corrected file system image is **error-free** when re-checked with your tool

### File System Layout

- **Block size: 4096 Bytes**
- **Total blocks: 64**
- **Block 0: Superblock**
- **Block 1: Inode bitmap**
- **Block 2: Data bitmap**
- **Blocks 3–7: Inode table (5 blocks)**
- **Blocks 8–63: Data blocks**
- **Inodes: 256 Bytes each**

## Superblock Structure

- **Magic Bytes: 2 Bytes (0xD34D)**
- **Block size: 4 Bytes**
- **Total number of blocks: 4 Bytes**
- **Inode bitmap block number: 4 Bytes**
- **Data bitmap block number: 4 Bytes**
- **Inode table start block number: 4 Bytes**
- **First data block number: 4 Bytes**
- **Inode size: 4 Bytes**
- **Inode count: 4 Bytes**
- **Reserved: 4058 Bytes**

## Inode Structure

- **Mode: 4 Bytes**
- **User ID of the file owner: 4 Bytes**
- **Group ID of the file owner: 4 Bytes**
- **File size in Bytes: 4 Bytes**
- **Last access time: 4 Bytes**
- **Creation time: 4 Bytes**
- **Last modification time: 4 Bytes**
- **Deletion time: 4 Bytes**
- **Number of hard links to this inode: 4 Bytes**
- **Number of data blocks allocated to the file: 4 Bytes**
- **Direct block pointers (point directly to data blocks): 4 Bytes**
- **Single Indirect block pointer: 4 Bytes**
- **Double Indirect block pointer: 4 Bytes**
- **Triple Indirect block pointer: 4 Bytes**
- **Reserved: 156 Bytes**

## Features

### 1. Superblock Validator

Verifies:

- a. Magic number (must be 0xd34d)
- b. Block size (must be 4096)
- c. Total number of blocks (must be 64)
- d. Validity of key block pointers: inode bitmap, data bitmap, inode table start, data block start
- e. Inode size (256) and count constraints

## 2. **Data Bitmap Consistency Checker**

Verifies:

- a. Every block marked used in the data bitmap is actually referenced by a valid inode
- b. Every block referenced by an inode is marked as used in the data bitmap

## 3. **Inode Bitmap Consistency Checker**

Verifies:

- a. Each bit set in the inode bitmap corresponds to a valid inode  
(Hint: An inode is valid if its number of link is greater than 0 and delete time is set to 0)
- b. Conversely, every such inode is marked as used in the bitmap

4. **Duplicate Checker** detects blocks referenced by multiple inodes

5. **Bad block checker** detects blocks with indices outside valid range

## **Mark Distribution**

<b>Features</b>	<b>Marks</b>
Superblock Validator	20
Data Bitmap Consistency Checker	20
Inode Bitmap Consistency Checker	20
Duplicate Checker	20
Bad block checker	20
<b>Total</b>	<b>100</b>

## Submission Guideline

- Submission guidelines can be found in the submission form. The link to the submission form is given below.

**[Submission Form Link]**

## Collaboration Policy

- This project is a group assignment. A group can consist of at most 3 people. The difficulty of the project will be adjusted according to the number of people in the group. Discussions are encouraged, but direct code sharing is prohibited.
- **Plagiarism will result in penalties according to university policies.**