1 File system consistency

Journaling **heavily reduces** the probability of having inconsistencies in a file system. In case of crash, the log **stores** what operations **were not run**. However, it can still be possible to get some **inconsistencies** (e.g. data blocks werent flushed to the drive, typical case on USB drives!). This can be problematic, in particular for **structural blocks** such as i-nodes, directories, and free lists. System utilities are available to restore file systems, e.g.: (Scandisk, FSCK) There are two main consistency checks: **block** and **directory**.

1.1 Block consistency

Block consistency checks whether blocks are **assigned/used** the correct way. Block consistency is checked by building two tables:

- Table one counts how often a **block** is present in a **file** (based on the i-nodes)
- Table two counts how often a **block** is present in the **free list**

A consistent file system has a 1 in either of the tables for each block. Typically, this is a **very slow** process, taking even hours (and running with the partition unmounted)

1.1.1 Restoring

- A **missing block**: it does not exist in any of the tables add it to the **free list**.
- A block is **double counted** in the free list (disaster waiting to happen) **re-build** the free list.
- A block is present in two or more files.
  - Removing one file results in the adding the block to the free list
  - Removing both files will result in a double entry in the free list
  - Solution: use new free block and copy the content (the file is still likely to be damaged)

FSCK Algorithm:

- Iterate through all the i-nodes - retrieve the blocks - increment the counters
- Iterate through the free list - increment counters for free blocks

1.2 Restoring I-node consistency

Checking the directory system: are the **i-node counts correct**? Where can it go wrong?:

- I-node counter is **higher** than the number of directories containing the file
  - Removing the file will reduce the i-node counter by 1.
  - Since the counter will remain larger than 1, the i-node / disk space **will not be released** for future use
- I-node counter is **less** than the number of directories containing the file
  - Removing the file will (**eventually**) set the i-node counter to 0 whilst the file is **still referenced**
  - The file / i-node will be released, even though the file was **still in use**

Recurse through the directory hierarchy - **Increment** file specific counters - I.e. each file is associated with one counter. One file may appear in multiple directories - Compare the file and i-node counters - Correct if necessary

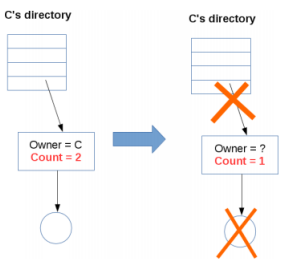

Figure: I-node counter is higher than the actual number of directories containing the file. Removing the file results in wasted memory.

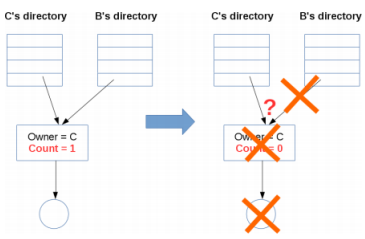

Figure: I-node counter is less than the actual number of directories containing the file. Removing the file results in a missing file.

2 File system

2.1 Defragmentation

At the beginning, all free disk space is in a **single contiguous** unit. After a while, creating and removing files, a disk may end up badly *fragmented* (holes and file all over the place). Defrag utilities make file blocks **contiguous** (very slow operation), and free space in one or more large **contiguous regions** on the disk. Windows users should run this regularly, except on SSDs. Linux (**ext2/3**) suffers less from fragmentation. Defragmentating SSD is **counter-productive** (No gain in performance and SSDs wear out).

2.2 History

- **Minix file system**: the maximum file size was 64MB and file names were limited to 14 characters.
- The extended file system (**extfs**): file names were 255 characters and the maximum file size was 2 GB.
- The **ext2** file system: larger files, larger file names, better performance.
- The **ext3-4** file system: journaling etc.

2.3 The extended 2 file system (ext2)

The second extended file system (**ext2**) is one of the **most popular** file systems in Linux. The main goals:

- Improve the performance of MINIX and extfs file systems, distributing directories evenly over the disk.
- Allow greater **file names** and **sizes**, improving directory implementation.

Figure: Standard Unix Partition

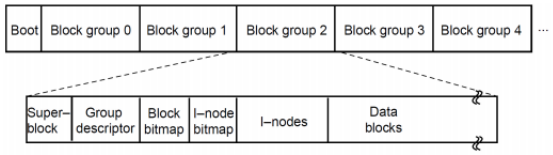

Figure: Ext2 Partition Layout (Tanenbaum)

An ext2 partition is split into several **block groups** to:

- **Reduce fragmentation** by storing i-nodes and files, and parent directories and files in the **same block** group if possible.
- **Reduce** seek times and **improve** performance
- All block groups have the **same size** and are stored **sequentially** (which allows direct indexing)

2.4 Directory entries

- The **superblock** contains file system information (e.g. the number of i-nodes, disk blocks)
- The **group descriptor** contains **bitmap locations**, the number of **free blocks**, **i-nodes** and **directories**
- A **data block bitmap** and **i-node bitmap**, used to keep track of free disk blocks and i-nodes (Unix uses lists)
- A **table** of i-nodes containing file and disk block information
- **Data blocks** containing file and directory blocks

Every directory entry contains the following fixed-length fields:

- i-node number
- Entry size in bytes
- Type field, i.e. file, directory, special file, etc.
- File name length in bytes
- And then, the file name itself (of variable-length).

Directories are searched **linearly** (i.e. they are **unsorted**) and a **cache is maintained** for recently accessed items.

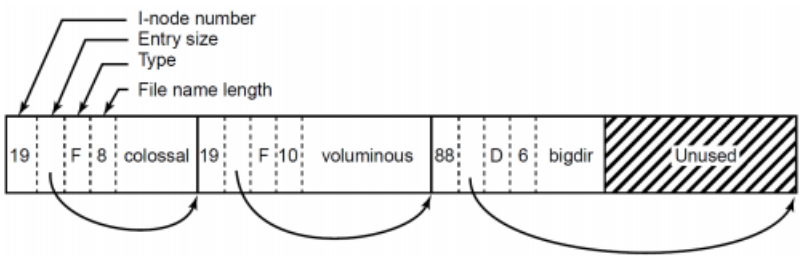

Figure: Ext2 Directory Implementation (Tanenbaum)

- File names up to **255 characters**
- **File lookups** are similar to the Unix file system.
- The i-node structure is similar to the Unix i-nodes
  - 12 block addresses are contained in the i-node
  - Single, double and triple indirect blocks are used
  - With a blocks of 1KB, this scheme can handle file sizes of 16GB.
  - If block size is 8KB, it could support file sizes up to 64TB.

2.5 Ext3

When making changes to an **Ext2** file system, files are ideally written immediately to prevent inconsistency: This generates **significant** head movement. Ext2 File system is more suitable for **flash disks** (no journal).
Ext3 builds upon the Ext2 file system by adding: **Tree based structures** for directory files to facilitate indexing (HTrees). **Journaling capabilities**

Reference section

placeholder