Project 3B

CS111 Discussion 1C

Note: You can form groups of 2 for this project

Note: Google ‘CS 111 samples’ for .csv’s

What is this project about?

- In P3A, input is a filesystem, output is a .csv
- Now : input is a .csv, which you have to analyze
  - Inconsistencies have been added to the filesystem, such as:
  - Allocated blocks/inodes on the freelist
  - Invalid data blocks
  - Allocated reserved blocks
  - Incorrect link counts to inodes ...
- This will be easier to do in Python, thanks to the csv library
  - If you use another programming language, you'll have to parse through the .csv yourself

Where to start?

- Whether you use Python or not, start by saving the .csv in a structure/class
  - The csv library in python makes this process easy
  - csv.reader() will iterate through lines for you
    - Make sure you handle empty lines/comments/trailing spaces
  - It will return a line of strings
    - You know that the first value determines what the rest of the line will be (eg: if first value is SUPERBLOCK, expect 7 more values)
- If using python, some useful methods for that class:
  - Recalculate #blocks/group and #inodes/group
  - Is a block number legal?
  - Is a block on the freelist?
  - Is an inode on the freelist?

I-node Allocation Audits

- Allocated i-nodes:
  - Should appear on the .csv
  - Should have a valid type
- Unallocated i-nodes:
  - Should not appear as an entry on the .csv, but should appear on the freelist
  - Will have a ‘zero’ type
- Compare the two sources of information (freelist and .csv entries) and report inconsistencies

- Remember that the first inodes are reserved by the filesystem. Therefore they won’t appear on the .csv or on the freelist, which isn’t an error.

Reserved inodes

Reserved ext2/ext3/ext4 inodes

So... What are the hidden and reserved inode numbers on ext2/3/4 filesystems?

Name	Number	Function
EXT2_BAD_INO	1	Allocated over bad blocks by mke2fs/e2fsck -c
EXT2_ROOT_INO	2	The root directory of the file system
EXT4_USR_QUOTA_INO	3	The now integrated and hidden user quota file. Previously used for ACL index?
EXT4_GRP_QUOTA_INO	4	The now integrated and hidden group quota file. Previously used for ACL data?
EXT2_BOOT_LOADER_INO	5	Apprently unused. Was probably intended to hide stage2 loaders, just like the fs starts padded with zeroes to allow stage1 loaders.
EXT2_UNDEL_DIR_INO	6	Apparently unused. Was supposedly meant for undeletion functionality that was never implemented.
EXT2_RESIZE_INO	7	Used to speed up resizing, by reserving room to let the block group descriptor table grow (mke2fs resize=) option.
EXT2_JOURNAL_INO	8	The ext3 journal (mke2fs -j)
EXT2_EXCLUDE_INO	9	Unused in vanilla, but used by the Next3 fs for snapshots.
EXT4_REPLICA_INO	10	Unused in vanilla, but used for ext4 metadata replication out of tree in a Google patch.

More junk by Vidar

These won’t be on the free list or in the .csv output, which is not an error

Block Consistency Audits

- Is the block reserved? On the freelist? Valid?
  - A ‘state’ variable can be used to track the block
  - Eg: state is invalid, reserved, free or in use.
- However, how will we know if a block is referenced several times?
  - Initialize the state to 0
  - Through a first pass, determine validity/free/reserved/usage and set the state accordingly
    - If the block you’re examining has its state set to in use
      - You’ve seen this block before!
      - Set its state to duplicate
  - Through a second pass, you can report on all instances of duplicate blocks!

Directory Consistency Audits

- Go through all directories
- You already scanned inodes
  - Check if the parent inode number is correct
    - Is it a valid inode number?
    - Is it an allocated inode?
- Make sure to count the links!
  - Keep a counter and iterate it
- Save the pointer to the parent
  - Through a second pass, make sure the parent is correct