5d. File System Checking

In order to generate broken file systems, we can modify mkfs to create file systems that fail one or more of the basic file system checks. Here are some ways to do this:

1. Superblock: Modify mkfs to create a file system with a corrupted superblock. For example, change the superblock signature to a value that does not match the file system type, or set the number of inodes to a value that exceeds the number of blocks available in the file system.
2. Block size: Modify mkfs to create a file system with an unsupported block size. For example, set the block size to a value that is smaller than the minimum block size, or set it to a value that is not a power of two.
3. Inode size: Modify mkfs to create a file system with an unsupported inode size. For example, set the inode size to a value that is smaller than the minimum inode size, or set it to a value that is not a multiple of the block size.
4. Number of inodes: Modify mkfs to create a file system with too few or too many inodes. For example, set the number of inodes to a value that is smaller than the number of reserved inodes, or set it to a value that exceeds the maximum number of inodes supported by the file system.
5. Number of blocks: Modify mkfs to create a file system with too few or too many blocks. For example, set the number of blocks to a value that is smaller than the number of reserved blocks, or set it to a value that exceeds the maximum number of blocks supported by the file system.
6. Block group descriptor: Modify mkfs to create a file system with a corrupted block group descriptor. For example, set the number of blocks in a block group to a value that exceeds the number of blocks available in the file system, or set the block bitmap block number to a value that is outside the block group.

By modifying mkfs to create file systems with one or more of these conditions, we can generate broken file systems that fail one or more of the basic file system checks. These file systems can

be used for testing purposes, to ensure that file system utilities and tools can handle error conditions correctly.

A basic file system checker in xv6 can be used to identify and correct inconsistencies in the file system. This can be useful for ensuring the file system's integrity, especially after a crash or power loss. Here's a simple implementation of a file system checker for xv6. Created a new C file called fsck.c inside the xv6 source directory. The fsck program will read the superblock and inodes from the file system image, checking for consistency. It will print messages for each check, indicating whether the superblock and inodes are consistent or corrupted.