Project 5a: File System Checker

Background

In this assignment, you will be developing a working **file system checker.** A checker reads in a file system image and makes sure that it is consistent. When it isn't, the checker takes steps to repair the problems it sees; however, you won't be doing any repairs to keep your life at the end of the semester a little simpler.

A Basic Checker

For this project, you will use the xv6 file system image as the basic image that you will be reading and checking. The file **include/fs.h** includes the basic structures you need to understand, including the superblock, on disk inode format (struct dinode), and directory entry format (struct dirent). The tool **tools/mkfs.c** will also be useful to look at, in order to see how an empty file-system image is created.

Much of this project will be puzzling out the exact on-disk format xv6 uses for its simple file system, and then writing checks to see if various parts of that structure are consistent. Thus, reading through mkfs and the file system code itself will help you understand how xv6 uses the bits in the image to record persistent information.

Your checker should read through the file system image and determine the consistency of a number of things, including the following. When one of these does not hold, print the error message (also shown below) and exit immediately.

- Each inode is either unallocated or one of the valid types (T_FILE, T_DIR, T_DEV). ERROR: **bad inode.**
- For in-use inodes, each address that is used by inode is valid (points to a valid datablock address within the image). Note: must check indirect blocks too, when they are in use. ERROR: **bad address in inode.**
- Root directory exists, and it is inode number 1. ERROR: **root directory does not exist.**
- Each directory contains **.** and **..** entries. ERROR: **directory not properly formatted.**
- Each **..** entry in directory refers to the proper parent inode, and parent inode points back to it. ERROR: **parent directory mismatch.**
- For in-use inodes, each address in use is also marked in use in the bitmap. ERROR: **address used by inode but marked free in bitmap.**
- For blocks marked in-use in bitmap, actually is in-use in an inode or indirect block somewhere. ERROR: **bitmap marks block in use but it is not in use.**
- For in-use inodes, any address in use is only used once. ERROR: **address used more than once.**
- For inodes marked used in inode table, must be referred to in at least one directory. ERROR: **inode marked use but not found in a directory.**

- For inode numbers referred to in a valid directory, actually marked in use in inode table. ERROR: **inode referred to in directory but marked free.**
- Reference counts (number of links) for regular files match the number of times file is referred to in directories (i.e., hard links work correctly). ERROR: **bad reference count for file.**
- No extra links allowed for directories (each directory only appears in one other directory). ERROR: **directory appears more than once in file system.**

Other Specifications

Your server program must be invoked exactly as follows:

```
prompt> fscheck file_system_image
```

The image file is a file that contains the file system image. If the file system image does not exist, you should print the error **image not found.** to stderr and exit with the error code of 1. If the checker detects one of the errors listed above, it should print the proper error to stderr and exit with error code 1. Otherwise, the checker should exit with return code of 0.

Hints

It may be worth looking into using mmap() for the project.

Make sure to look at **fs.img** , which is a file system image created when you make xv6 by the tool **mkfs** (found in the tools/ directory of xv6). The output of this tool is the file **fs.img** and it is a consistent file-system image. The tests, of course, will put inconsistencies into this image, but your tool should work over a consistent image as well. Study mkfs and its output to begin to make progress on this project.