CMU 18-746 | **Storage Systems** | **Assigned: 18 Feb 2008**
Spring 2008 | **Project 2: fsck** | **Due: 7 Mar 2008**

Overview: In this laboratory you will create tools to identify, parse, read, and manipulate the on-disk image of a file system. The end product of this project is a rudimentary `fsck` utility for the ext2 file system.

Many file systems contain a `fsck` (file system check) utility to check for and repair errors in the file system at mount time. Before starting work on this laboratory, read *Fsck—the UNIX file system check program* (McKusick & Kowalski 1994, revised 1996). Although this paper refers to the BSD FFS `fsck` utility, most of the concepts hold valid for ext2fs and many other file systems.

You will do this lab on your own. While you can talk with others about the project at a high level, you must do all the work, and write all the code by yourself.

The three steps below outline the work that must be completed for this project. The steps are not independent; each of Steps II and III depends on the tool created in the previous step.

Step I: Read the partition table

You have been provided with a disk image (see *Resources* below). Build a tool to read and *print out* both the DOS-style partition table located on sector 0, and the extended partition information (pointed to by partition 4) for extended partitions 5 and 6. You can verify that you've done this correctly when your code produces the following values:

partition number	partition type	start	length
1	0x83 (ext2)	63	48132
2	0x00 (unused)	0	0
3	0x83 (ext2)	48195	48195
4	0x05 (Extended)	96390	64260
5	0x82 (Linux swap)	96453	16002
6	0x83 (ext2)	112518	48132

Partition 1 contains an ext2 file system with no errors. Partitions 3 and 6 contain ext2 file systems with various errors that should be detectable and correctable by your utility. The files and directory structure on each partition table are similar but not necessarily the same.

Step II: Read the file system structures

Extend your tool from the previous section to accomplish the following items. In general, each item depends on the functionality of the items above it.

- Read the ext2 superblock for partition 1. This is located at offset 1024 bytes into the partition (not the disk), and is of size 1024 bytes. *Print out* the superblock magic number and verify that it is is correct.
- Determine how to translate an inode number into its equivalent sector number. To do this you will need to read the correct entry in the inode table—which itself is split across multiple block groups. (The first inode in the inode table is inode number 1, not 0.)
- Determine how to locate an inode's (and block's) entry in the inode allocation bitmap and block allocation bitmap.
- Locate and read the root inode. Verify that the inode attributes show that it is a directory inode. Verify that this inode is allocated in the inode allocation bitmap.
- Read the directory information pointed to by the root inode. (Be sure to use `struct ext2_dir_entry_2` for this.) Determine the directory entry that points to the directory `/lions`.

- Read the inode for the directory `/lions` and *print out* the inode number. This should be inode 4017. Verify the attributes of this inode. Continue this process until you find the file `/lions/tigers/bears/ohmy.txt`. *Print out* the inode number of this file – it should be 4021. Verify that the data blocks for this file are allocated in the block allocation bitmap.
- *Print out* the inode number for the file `/oz/tornado/dorothy`. What is special about this file?
- Print out the inode number for the file `/oz/tornado/glinda`. Verify that this file's type is a symbolic link. *Print out* the name of the file this link references.

Step III: Check for and repair any file system errors

Extend your tool to checks for the specific file system errors listed below. Your tool should make four "passes", checking for the specified errors in each pass. When an error is found, you should print a description of the error to `stdout` and automatically fix the error. Your tool must only generate output when detecting and repairing errors—in other words, it should generate no output for a correct file system.

- **Pass 1: Directory pointers** (see McKusick & Kowalski, section 3.7). Verify for each directory: that the first directory entry is "." and self-references, and that the second directory entry is ".." and references its parent inode. If you find an error, notify the user and correct the entry.
- **Pass 2: Unreferenced inodes** (section 3.5). Check to make sure all allocated inodes are referenced in a directory entry somewhere. If you find an unreferenced inode, place it in the `/lost+found` directory—make the new filename the same as the inode number. (I.e., if the unreferenced inode is #1074, make it the file or directory `/lost+found/#1074`.)
- **Pass 3: Inode link count** (section 3.5). Count the number of directory entries that point to each inode (e.g., the number of hard links) and compare that to the inode link counter. If you find a discrepancy, notify the user and update the inode link counter.
- **Pass 4: Block allocation bitmap** (section 3.3). Walk the directory tree and verify that the block bitmap is correct. If you find a block that should (or should not) be marked in the bitmap, notify the user and correct the bitmap.

You may run your tool against the file systems on partitions 3 and 6 as a test. You should find one of each error (two on one file system, two on the other). Your program will be evaluated by running it against a similar file system and determining whether it outputs a error-free file system.

Deliverables: All deliverables are due at the beginning of class on the assigned date. Late submissions will not be accepted, and no extensions will be granted.

You will email a `.tar` file to your friendly T.A. This file will contain the source code for your tool and a `Makefile` to build your tool. Be sure to include all header files and any supplementary files required for the `make`. The executable you generate will be called `myfsck`, and should take one argument—the name of the "file" containing the disk image to be checked. In order words, your friendly T.A. must be able to do this:

```
unix49{~}% mkdir foo ; cd foo
unix49{foo}% tar xf ../myfsck.tar
unix49{foo}% make
unix49{foo}% ./myfsck ../some_disk_image
```

Your tool **must** compile on the Linux systems at `unix.andrew.cmu.edu`. We will test your code by running your fsck program on the partition, and then running standard fsck to make sure you caught and corrected all the appropriate errors.

In your email, please tell your friendly T.A. what you like and dislike about this laboratory, and whether there is anything you would suggest we change (to make it easier to understand, more challenging, etc.)

Resources: The following resources are available on the course web site:

- `genhd.h` and `ext2_fs.h`: These are relevant header files from the Linux 2.4.17 kernel distribution. These files contain the structures you will need to read in order to interpret the on-disk file system organization. You will not need to use all the structures and `#defines` from these files.
- `readwrite.c`: This is a stub program that will read and write one "sector" at at time for you from the disk file. You should use this program as the basis for your `fsck` tool.
- `disk`: This is a disk image that contains 5 partitions. You should use this disk for your tool development and testing. Assume that the disk sector size is 512 bytes, and that sector numbering starts at 0.

You should use the Internet (and the comments in the Linux header files) to figure out how to interpret the on-disk data structures. Here are some suggested starting points (this list is neither exhaustive nor necessarily the best sources of information):

Information on partition tables:

`http://www.tldp.org/HOWTO/Large-Disk-HOWTO-6.html`
`http://www.tldp.org/HOWTO/Large-Disk-HOWTO-13.html`

Information on ext2 file system internals:

`http://www.tldp.org/LDP/tlk/fs/filesystem.html`
`http://homepage.smc.edu/morgan_david/cs40/analyze-ext2.htm`
`http://uranus.it.swin.edu.au/ jn/explore2fs/e2fs.htm`