CS 5600 – Computer Systems
Homework 4 – File System Implementation

For details on the cs5600fs file system format and the FUSE library please see the accompanying document.

Materials

You will be provided with the following files in your <id>-hw4 SVN repository:

- compile.sh
- homework.c – skeleton code
- misc.c – additional support code
- image.c, blkdev.h – the disk image device and blkdev header file (see docs)
- disk1.img.orig – sample disk image
- mkfs-hw3.c – utility for creating new disk images
- read-img.c – utility for parsing disk image files and displaying them.

Additional information beyond the accompanying document and README.txt may be found in source file comments.

Question 1 – command line read-only access

Implement code for read-only command-line access to 5600fs disk images. (note that this is the same as the code for read-only FUSE access; however you will debug and test with the command-line interface, which is simpler and easier to debug)

Suggestions:

- At startup create a copy in memory of the superblock and the file access table; access these instead of going to disk for that information.
- For splitting paths into components you may wish to use the 'strwrd' function from the c-programming.pdf file. (under “Course materials” in Blackboard)

Testing:

You will be responsible for writing a test script which verifies your code; please provide a copy of this script named 'q1test.sh'. You are provided with an executable file 'q1-soln' which may be assumed to implement these functions correctly; you may use this in your test process to provide output you can test against.

Cases which you should test:

- 'ls' returns the correct output in all directories of disk1.img
- 'ls-l' returns correct output for selected files
- statfs returns the correct output
- The 'blksiz' command sets the size of the buffer passed in the read() call to your filesystem. Verify that your code works correctly for multiple values of this parameter – suggested values (besides the default 1000 bytes) are 17, 1024, and 4000 bytes.

Note that in writing your test script it may be useful to use a shell programming feature which allows you to specify the input to a command within the script:

```
./homework --cmdline disk1.img <<EOF
ls
cd home
EOF
```

The lines between '<<EOF' and 'EOF' (or other identifier, but EOF is traditional) will be used as standard input to the command (./homework) when it is run.

Question 2 – command line read/write access

For this question you will need to implement read/write access through the command line interface.

Suggestions:

- Track changes to the file access table and flush them to disk at the end of each operation.
- Use the read-img utility to check whether you are writing to the disk image correctly. Make sure you don't modify the original disk image file (setting it read-only helps) and be aware that if buggy code corrupted the image you are using, things may still fail after you fix the bugs unless you start again with a fresh image.

Testing:
Provide a script file named 'q2test.sh' which tests the following functionality, which your code should pass:

- create a directory, 'cd' into it, create a file, have it show up in 'ls', remove the file, verify it's gone, 'cd ..', and remove the directory.
- Verify that you can create short (<1024 bytes) and long (> 2048 bytes) files, and that they contain the correct content.
- Verify that you can nest directories, create and remove files in the nested directories, and remove a subdirectory without messing up the root directory.
- Re-run your file creation tests with the same values of 'blksiz' you used for question 1.
- Verify that you cannot remove a directory with files or directories in it.

Question 3 – FUSE access

Test and debug FUSE access.

Suggestion: The first thing that FUSE does after mounting a file system is to call getattr(“/”), so you need to handle 'getattr' properly for the root directory, which has a corresponding dirent in the superblock.

Testing – you will need to test the same functionality as for questions 1 and 2, plus file attributes such as creation time and permissions. Please see the file q3test-example.sh for an example of techniques which may be helpful in performing these tests.