Background

This assignment continues your work on the FAT-based file system and provides a rudimentary way to inspect a file system. Unlike the earlier parts of this assignment, you will be given more latitude in the design of your program — so you should take the opportunity to think of how to structure your program accordingly. Use appropriate functions, parameters, and data structures.

Extra Credit

This assignment includes an optional exercise, available for those seeking extra credit. If you are interested in this, please contact me before starting your work.

Exercise 1. Design and implement a primitive command-line interpreter, named `interp`, capable of acting on the following two commands:

- `dir`
- `cat`

The `dir` command should output a directory listing of the root directory of the diskette image, in the spirit of the early DOS operating system. The output should include file attributes, date and time stamps, file name, and file size. For example, directory output should appear in a format similar to the following:

```
-D---- 11/18/02 08:37 TEXT         0
A----R 11/18/02 08:14 LAZY1   TXT 46
A----- 11/18/02 08:21 LAZY10  TXT 738
```

The `cat` command takes one argument: the name of a file, such as `lazy10.txt`. In response, the contents of the file should be displayed on the terminal. File names are case-insensitive, so `lazy10.txt`, `LAZY10.TXT` and `lAzY10.tXt` all refer to the same file.

To display files, it is important to be aware of a small detail regarding the FAT 12 system. Recall that entries 0 and 1 of the FAT are reserved. As a result, the sector values associated with files are integer values no smaller than 2. This basic fact applies both to the start sector values stored within the directory entries and also to the values which appear within the sector chains stored in the file allocation table. Recall that for a 1.44 Mb diskette the data sectors are numbered from 33 to 2879. A fixed offset of 31 is thus involved—the sector numbered 2 in the FAT actually refers to sector 33, the sector numbered 3 in the FAT refers to sector 34, and so forth.

Modify your `Makefile` so that it supports a new target, named `interp`.

Exercise 2 (Extra Credit). In the earliest DOS file systems, no concept of subdirectories existed. (This oversight was corrected with the introduction of DOS 2.0, when dinosaurs roamed the earth.) This allows for a relatively simple implementation, but the result is inconvenient to end-users, especially with the 8+3 character limit on file names. The goal for this part of the assignment is to add support for subdirectories.

In the FAT 12 system, there are some notable differences between the root directory and subdirectories. These differences are due to the history of DOS, and they will probably have an impact on how you go about writing your code.

These differences include the following:

- For the root directory, the sectors are contiguous. For subdirectories, the sectors need not be contiguous.
- The root directory has a fixed number of sectors. Subdirectories are implemented like files and can thus vary in size.
- For the root directory, the sectors are known to lie between 19 and 32. For subdirectories, the sectors are obtained from the FAT and the application of a fixed offset.
- Subdirectories include explicit **.** and **..** entries; the root directory does not.

To get a good understanding of how things are stored in subdirectories, you can inspect the provided diskette image. In particular, pay special attention to the **.** and **..** entries and their "start sector" values. For example, inspect the sectors of the diskette corresponding to **/**, **/ctan**, and **/ctan/eso-pic**.

Design and implement a new command, `cd`, which will allow the user to specify a subdirectory—either by name, or using the Unix-inspired `.` and `..` syntax. Subsequent `dir` and `cat` commands should work as before, relative to the current directory.

What to Submit

Before you submit your work, verify that your program compiles and works properly in the Mac OS X environment. Your `Makefile` should, at a minimum, support the following targets:

```
make clean
make interp
```