CprE 308 Laboratory 8: The FAT-12 Filesystem

Department of Electrical and Computer Engineering
Iowa State University

1 Submission

Turn in your well-formatted and commented source code and a copy of the output.

2 Introduction

The purpose of this lab is to gain additional experience working with filesystems. This is an extension of the previous lab on the FAT-12 filesystem. You will implement a program simiar to "ls" in unix or "dir" in DOS.

2.1 Terms

FAT (File allocation table) a linked list of clusters, represented within a table. On FAT-based filesystems, there are two copies of this structure to provide redundant information in the event of corruption.

Root directory a list of files and directories. The root directory is a sequence of entry, with one entry for every file or directory within the root directory. For each file, we can find vital statistics about the file, including the size, time of creation, as well its starting cluster. We find the remaining clusters of a file by following the linked list in the FAT.

Cluster a group of sectors, generally in a power of 2 (1, 2, 4, 8, 16 or 32 in FAT-12). The data area is carved up into clusters; the number of sectors/cluster determines what the resolution is. More sectors/cluster means you can address a larger data area, but you potentially waste more space due to internal fragmentation with smaller files.

2.2 Overall Floppy Disk Layout

Boot Sector	FAT	2nd FAT	Root Directory	Data
0	1	X	Y	Z

Floppy disks have logical sectors starting at 0. This is where you'll find the boot sector. The first FAT always starts at logical sector 1, but after that you have to calculate the starting positions using information from the boot sector. The formulas for finding the sector numbers for the 2nd FAT, the root directory, and the data area are as follows:
$X = 1 + sectors/FAT$
$Y = X + sectors/FAT$
$Z = Y + roundup512(32 \cdot directoryEntries)$ /* have to start at the next open block */
Note: roundup512() rounds up to the next multiple of 512; this is because data must start at an open block The last formula is because each directory entry takes 32 bytes of space.

2.3 Root Directory Layout

The root directory is an array of directory entries (the number of such entries is given in the boot sector). Each directory entry is 32 bytes and is laid out according to Table 1.

Offset	Length	Description
0x00	8	Filename
0x08	3	Extension
0x0b	1	Attributes
0x0c	10	Reserved
0x16	2	Time
0x18	2	Date
0x1a	2	First Cluster
0x1c	4	Size

Table 1: Directory Entry

Notes on interpreting a directory entry:

If the filename starts with 0x00, it's not a valid entry. If it starts with `0xE5`, it's also an invalid entry, but it has been released/deleted. If it starts with anything else, assume that it's a valid file entry.

Filename in DOS, the naming convention is 8.3. That is, files have 8 character names and 3 character extensions, such as `FILENAME.EXT`. We will not cover the long filename extensions in this lab.

Attributes a file's attributes include read/write permissions, among other things. There are 8 bits, but we're only interested in a few of them for this lab. If a bit is set, it indicates the property is true. 0 is the least significant (rightmost) bit.

Bit	Attribute
0	Read-only
1	Hidden
2	System
5	Archive

Date and time These fields are each 16 bits long. The bits are laid out as follows:

The time is encoded using 5 bits for the hour, 6 bits for the minutes, and 5 bits for the seconds. One catch - Since 5 bits is not enough for 60 seconds, only every other second is counted. In other words, if your counter is 16, that indicates a value of 32 seconds.

The date is encoded in a yyyy/mm/dd format. It uses 7 bits for the years since 1980, so add 1980 to the counter to get the current year. There are 4 bits for the month, and 5 bits for the day.

3 Your Program

Write a function that does what `ls` would do: print out the filename, time, date, attributes, and size of the file for each valid entry in the root directory. No need to sort the files – simply list them in disk order.

Once you have fields from the boot sector, you can calculate the starting sector for the root directory. There you should find the array of directory entries. The boot sector also tells you how many root directory entries the filesystem supports, so you can loop through a finite number of potential entries.

Call your program `fat12ls.c`. Please use the provided C program template (fat12ls-template.c) to create your own program.

3.1 Example Output

Your output might resemble the following example:

```
Name BIG.LOG
Attrib RHS
Date/time 21:37:24 2002/11/06
Size 78870 bytes
```

Note that these values my not be the correct ones you will get from the image. Feel free to improve on the format.

Your program should be as modular as possible. Encapsulate operations inside functions or macros. We will award some portion of the points for comments, readability, and modularity. The style guide from the previous lab still applies!

3.2 Tips

Some suggestions for breaking the problem down:

- have a function to find the directory entries and loop through them
- have a function for decoding valid directory entries
- have separate functions for decoding complex fields (attributes, date, time, etc)

3.3 Grading Criteria

TAs will grade the project using the following criteria:

- Summary and Report: 10 points
- Program: 16 points
- Correct output: 24 points