The FAT-12 Filesystem

April 12, 2016

1 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 similar to `ls` in Unix or `dir` in DOS. As in the previous lab, you may find this website very useful as you work on the lab.

1.1 Terms

1.1.1 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.

1.1.2 Root directory

A list of files and directories. The root directory is a sequence of entries, 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 as its starting cluster. We find the remaining clusters of a file by following the linked list in the FAT.

1.1.3 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 divided 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.

1.2 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 + *roundup*512(32 * *directoryEntries*) [have to start at the next open block]

Note: *roundup*512() rounds up to the next multiple of 512; this is because data must start at an open block. The formula in the parentheses is due to the fact that each directory entry takes 32 bytes of space.

1.3 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 the table below.

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

Notes on interpreting a directory entry: If the filename starts with 0x00, or 0xE5, it is not a valid entry. Filenames starting with 0xE5 are entries that have been released/deleted. If the filename starts with anything else, assume that it is a valid entry.

Filename In DOS, the naming convention is 8.3; meaning that files and directories have names that are 8 characters long, and extensions that are 3 characters long, such as `FILENAME.EXT`. If a filename is longer than 8 characters, then it is represented using multiple entries. Long file name entries always have a regular 8.3 entry to which they belong, and they immediately precede that 8.3 entry. More information regarding long file names can be found here.

Attributes A file's attributes include read/write permissions, whether or not a file is hidden, among other things. There are 8 bits, and if the bit is set, it indicates the property is true. The bits represent the following properties:

Bit	Attribute
0	Read only
1	Hidden
2	System
3	Volume ID
4	Directory
5	Archive

Note: If the entry is a long file name entry, the attribute byte will be set to `0x0F`.

Also of note, bit 0 is the least significant bit(i.e., the right most bit).

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

- **Time**:
  - 5 bits for the hour
  - 6 bits for the minutes
  - 5 bits for the seconds
    - Note that 5 bits is not enough to represent 60 seconds, so only every other second is counted. This means that if your seconds count is 16, that indicates a value of 32 seconds.
- **Date**: encoded in a yyyy/mm/dd format
  - 7 bits for the years since 1980, so add 1980 to the value to get the current year
  - 4 bits for the month
  - 5 bits for the day

2 Task for this lab

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. This program should also support a `-R` flag, which will tell the program to recursively traverse the directories and print out the information for the files in the different directories as well. No need to sort the files in each directory - simply list them in disk order. Make sure to state the directory that is being printed before printing out the contents.

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.

You may use your program from the previous lab as a starting point.

2.1 Example Output

Your output might be in the following format:

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

3 Extra credit

As it was in the previous lab, extra credit will be given for supporting the reading and listing of the files and directories in the FAT-16 and FAT-32 filesystems.

4 License