CS342 Spring 2015 - Project 5
Demystifying FAT32 File System

Assigned: 02.05.2013
Due date: 15.05.2013, Friday, 11:55 pm

In this project you will explore the FAT32 file system. You will learn about that, create a FAT32 file system, and analyze it using a tool that you will develop. Project will be done in Linux and using C programming language.

First read and learn about FAT32 file system (its on-disk structures, directory structures, FAT table structure, etc.). You can find a lot of documentation in Internet ([2], [3]).

Now, create a regular file (binary file) that will act as your disk (virtual disk). It will be initialized to all zero. You can do this by using the dd command in Linux.

```
$ dd if=/dev/zero of=disk.img bs=1k count=100000
```

This will create a file disk.img that will have a size that is worth of 100000 blocks where each block is 1 KB. That means the file size will be around 100 MB.

Use the **losetup** command to associate a loop device with the file (making it look like a block device instead of just a regular file within the file system) [1].

```
$ losetup /dev/loop0 file.img
```

Each device in a Linux machine has a /dev/… file associated with it. For example, your hard disk may have a /dev/ file called /dev/sda1 (or something like that), which can be used to access the hard disk in raw mode, as if you are accessing a binary file (open, read, etc). Now you can refer to your disk with the corresponding /dev/loop0 file. In a program, for example, you can open the /dev/loop0 program and read bytes from it.

With the file disk.img (your disk) now appearing as a block device (represented by /dev/loop0), like a hard-disk, create a FAT32 file system on the device (this is formatting – logical formatting). You can use **mkdosfs** command for this purpose. Read the man page carefully. There can be some other tools that you can use, instead of mkdosfs, for example, check the **mkfs** command. You need to specify the size of the file system, for example 10000 blocks, and also blocksize, for example 1 KB (or 1024). Read the manual page (or Internet documents) carefully.

Then create a mount point (create an empty directory with mkdir), such as /mnt/mydisk, and mount the new file system that you created to this mount point. You may need to use the sudo command. Use the **mount** command for this purpose. Read the man page of the mount. You may need to specify the type of the file system (the fact that it is a FAT32 file system).

After mount is successful, you can now change into the new new file sytem's root directory. To do that type: cd /mnt/mydisk. You will be in directory /mnt/mydisk. Type ls now. You will see such an output:

```
$ lost+found
```

Hence, there will be only one file there (lost+found) in the root directory (i.e., in /mnt/mydisk) of your file system. You can now create new directories and files, as many as you wish. You can create folders in a folder, or files in a folder (directory).

Create two files x and y in the /mnt/mydisk (root) directory. You can copy these files from somewhere else.

You can unmount the file system using the command **umount**.

Now unmount it.

You can now analyze (read byte-by-byte or block-by-block) the disk (i.e., the /dev/loop0 file or the disk.img file). Write a tool to do that. The tool C file will be called **fat32tool.c**. Hence, the executable will be fat32tool.

It will take a filename (corresponding to a file) as a command line parameter. It will parse the /dev/loop0 file (disk), i.e., it will read the disk in raw mode (first open the /dev/loop0 with open() function and then read with read() function; check the manual pages of open and read system calls – standard library functions). Your program will find out which blocks of the disk (virtual disk) are allocated to the file. This will require your program fat32tool.c to parse the FAT table included in the disk (/dev/loop0 or disk.img) and find out the block numbers of the data blocks used by the file. An example invocation of the program can be like the following:

```
fat32tool x
```

Output can be nothing if file x has a size 0. Output can be as below if file x has been allocated blocks 200, 2001, and 305 (3 blocks), for example.

```
        200
        2001
        305
```

If no filename is given as a parameter to your program, then your program will *list* the root directory. That means the names of the files and subfolders in the root folder of your filesystem will be printed out to the screen.

Submit a report, PDF file (report.pdf), explaining about how you have done the project. Include also the source code (look.c) in the report. Submit also look.c. Hence you will submit two files: report.pdf and look.c.

You can include more functionality in your look.c if you wish.

You will do a demo.

References:

1. Anatomy of the Linux File system,
http://www.ibm.com/developerworks/linux/library/l-linux-filesystem/
2. FAT32 File Systems Specificaton: https://msdn.microsoft.com/en-us/library/gg463080.aspx
3. FAT file systems: http://wiki.osdev.org/FAT