Operating Systems

P. Geens – F. Sanen – E. Steegmans

Manual analisys of the FAT12 file system

The purpose of this assignment is to analyse a FAT file system manually, only by using a hex editor. Most recent filesystems are very complicated. FAT12 is the first version of the FAT filesystem, and thus the simplest (it´s successor is FAT32). FAT12 is still widely used on floppy disks.

Not the whole FAT12 structure is explained in this document, for this purpose other interesting resources exist:
http://www.maverick-os.dk/FileSystemFormats/FAT12_FileSystem.html
http://www.ntfs.com/fat-systems.htm
http://en.wikipedia.org/wiki/File_Allocation_Table

There are many hex editors available, but most of them are commercial, and they have features like 'solve this lab automatically in 1 second', so obviously we are not going to use them. A very basic hex editor is the free version of the cygnus hexeditor (http://www.softcircuits.com/cygnus/fe/), which also is available on Programs.

FAT12 is a file system mostly used on floppy disks. To be able to work with them easily at the byte level, especially under Windows, we make an exact image. Under Linux dd is the way to go, under Windows you can use rawrite (http://uranus.it.swin.edu.au/~jn/linux/rawwrite.htm).

Do not forget to verify the image, to be certain it is an exact replica of the original. In forensics this is extremely important, so one can check if the image is altered. A way to achieve this is by taking an MD5 hash of both the original media and the image. When both hashes are the image is forensically correct. Under Linux md5sum is available, amongst many others. Also under windows there are several tools up for the job. Check. http://www.irnis.net/gloss/md5sum-windows.shtml or Google for hksfv.

If you would take a forensically exact image of the floppy and name it fat12.img, the resulting MD5 should be 537d30df6079d9d785d6a7e4cff1a786. Feel free to test it!

Creating a floppy disk for every student is a time consuming job, so we created the image fat12.img in advance. But do verify the MD5 checksum!

Floppy Disk

The structure of a FAT12 formatted floppy disk is as follows:

Position	Length	Data
0	1	Boot Sector
1	9	Fat 1
10	9	Fat 2
19	14	Root Directory
33	2847	Data

The sectorsize on a FAT formatted floppy is 512B. With unknown FAT-filesystems, parameters like sectorsize, number of sectors in the root-directory, and others need to be read from the BIOS parameterblock in the bootsector.

Task 1: *Taking into account that the sector size on a floppy is 512B, you should be able to isolate these different parts for the floppy image (fat12.img) by using a hexeditor. Find the start addresses for every part of the structure.*

Directories

Every entry in a FAT12 directory contains 32 bytes.

Offset	Length	DATA
0	8	Name
8	3	Extension
11	1	Attributes
12	10	Reserved
22	2	Time
24	2	Date
26	2	First cluster
28	4	Filesize

As an example take the following 32 bytes:

```
5445 5354 2020 2020 4444 2020 0064 851b
5a33 5a33 0000 851b 5a33 0300 b004 0000
```

Interpreting these as a directory in the FAT12 filesystem gives us the following results.

Name & Extension

The first 8 bytes contain the filename. With a little help from an ASCII table we can 'translate' these hexadecimal values: '`54 45 53 54 20 20 20 20`' becomes 'test '. The extension '`44 44 20`' will be 'dd '. Spaces are used as padding, so the actual filename is 'test.dd'.

Attributes

The attributes are kept in a bitvector. The meaning of the individual bits is as follows:

0:	read-only	4:	subdir
1:	hidden	5:	archive
2:	systemfile	6:	/
3:	volume label	7:	/

20 is the hexadecimal representation of 32 or '00100000'. Do not forget these fields are saved 'little endian', so bit 7 is left and bit 0 right. Only the archive bit is 1 for this file.

Time and Date

The 16 bits reserved for time are distributed as follows: 5 for the hours, 6 for the minutes, and the last 5 for the seconds. With 2_5 being only 32, only even values are used for seconds. The resulting values need to be doubled.

The binary representation of 1b85 is 0001101110000101. This results in 00011 hours (3), 011100 minutes (28) and 2 x 00101 (2 x 5) seconds. From the 16 date bits there are 7 available for the amount of years since 1980, 4 for the month and 5 for the day. This gives us 26/10/2005.

First cluster

The sequence number of the first cluster of the file is saved here. In our case it´s cluster 3, because 0300 should be read as 0003. If the file were empty this value should be 0.

Filesize

'`b004 0000`' becomes 04b0. Thus the filesize is 1200 bytes.

Task 2: *Interpret the following directory entry:*

```
    5a57 4152 544b 4153 584c 5310 0000 3633
    5a33 5a33 0000 3633 5a33 4001 0000 0000
```

Task 3: *Visualize the contents of the root directory of fat12.img. Give name, size and date of each entry.*

FAT-12

If we can find the starting cluster of a file in the directory entry, we can find all other clusters possibly used by this file by reading the FAT (File Allocation Table). For every cluster a 12 bits entry is reserved in the FAT.

The content of these entries:

`000`:	free cluster
`002-FEF`:	cluster in use, value points to the next cluster of the file
`FF0-FF6`:	reserved
`FF7`:	bad cluster
`FF8-FFF`:	cluster in use, this is the last cluster of the file

As a fat entry is 12 bits; 2 fat entries fit in 3 bytes. If we read from the FAT the bytes UV WX YZ, the last 2 FAT entries are XUV en YZW. Pay attention to the order of the bits. Remember, these are stored Little Endian.

Consider the first 12 bytes of the FAT are as follows:

```
F0 FF FF 00 40 00 05 F0 FF 00 00 00.
```

We can rewrite this as:

```
(F0F FFF) 000 004 005 FFF 000 000
```

Clusters 3, 4 and 5 are in use, cluster 2 is free. This FAT is taken from the directory example above, in which the first cluster pointed to 003. Here 003 points to 004, 004 to 005 and 'FFF' in the entry of cluster 005 tells us that this is the last cluster of the file.
The file is thus written in clusters 3,4 and 5. The directory told us the filesize is 1200 bytes, quite logical this file´s data is written in 3 clusters of 512 bytes. The remaining bytes in cluster 5 is:

(3 x 512 B) – 1200 B = 336 B

These 336 Bytes are called slack space. It is lost space due to internal fragmentation.

Task 4: *Find all clusters of the file som.xls on fat12.img and reconstruct the file.*

Removing files

Extra: *Find out what happens when a file is removed. How can you see this on the floppy? Is it possible to undelete a deleted file? How? If yes, are there limitations?*