ECE 598 – Advanced Operating Systems
Lecture 17

Vince Weaver

http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

31 March 2016

Announcements

- Homework #7 will be posted
- Project topics were due, should have received e-mail

Fat FS

- FAT-12/FAT-16/FAT-32
- Various block sizes from 512 - 32kB (tradeoffs)

Overall Format

offset	description
0	Boot Block
512	Fat #1
...	
...	Fat #N
...	Root Directory
...	Data Blocks

Boot Block

512 bytes, first part configuration info (block size, blocks in disk, FATs, etc), rest actual boot loader code

Offset	Length	Description
0x00	3	bootstrap (jmp to later)
0x03	8	manufactuer desc
0x0b	2	bytes per block
0x0d	1	blocks per unit
0x0e	2	reserved blocks (usu. 1 for boot block)
0x10	1	number of FATs
0x11	2	total root dir entries
0x13	2	blocks per disk. if ¿ 2**16 see 0x20
0x15	1	media descriptor
0x16	2	FAT size (blocks)
0x18	2	blocks per track
0x1a	2	disk heads
0x1c	4	hidden blocks (usually 0)
0x20	4	blocks on entire disk
0x24	2	drive num
0x26	1	boot signature
0x27	4	volume serial number
0x2b	11	volume label
0x36	8	fs id
0x3e	0x1c0	rest of boot code
0x1fe	2	0x55aa (end of boot block)

File Allocation Table (FAT)

One or more copies of File Allocation Table (FAT). Why multiple copies? Actually has to fit entirely in RAM. Just a table of 16-bit values, one for each cluster pointing to the next cluster in the file.

Entry 0 and 1 are reserved. 0 holds id, 1 holds the end-of-chain marker (usually 0xffff) The last entry in a list is 0xffff.

0 means unused. 1 reserved. 0xfff7 might mean bad cluster.

Size of entry 12=fat12 (3 bytes hold 2 cluster) 16 fat16, 32 fat 32

Example, a file might start at 2:

offset	value
0	//////
1	//////
2	3
3	5
4	0
5	ffff
...	
N	0

Root Directory

How do we know where a file starts? Root directory entry follows after last FAT.

Values are little endian

offset	size	description
0x00	8	filename
0x08	3	extension
0x0b	1	attributes
0x0c	10	reserved
0x16	2	creation/update time (h/m/s) second must be even
0x17	2	creation/update date
0x1a	2	start cluster
0x1c	4	filesize (bytes)

- Filename: First byte 0x0 = never used, 0xe5 = file deleted (sigma) (how can you undelete? restore first char, then hope the file was contiguous and restore as many clusters as the filesize says), 0x05 first char actually 0xe5, 0x2e this is current directory. If another 0x2e '.' then cluster field is parent directory (..) 0x00 means root If not 8 chars, padded with spaces

- Extension: three bytes. dot is assumed

- Attributes: 0x1=r/o, 0x2=hidden, 0x4=system, 0x8=disklabel 0x10 subdirectory, 0x20=archive (for backups)

- Time: hhhhhmmmmmmsssss. seconds has to be even
- Date yyyyyyymmmmddddd y $= 0$-199 (1980-2099)
- Directories: if attribute set, then cluster chain treated as a series of directory entries

Other Fat info

- Cluster size. Have to make it bigger to fit filesystems bigger than 32MB. Why can that be bad? (mostly, wasted space with small files)

- VFAT – long filenames and others, win98.
  Says there is one with invalid attr value 0xf
  A dummy file entry is put beforehand to hold long name.
  ALso a compatible one is created.
  Also reserved 10 bytes, extend file time to have ms resolution, extra timestamps.

- FAT32 – increased sizes so can have max filsesize by 4GB.
- UMSDOS – linux filesystem that let you have permissions, long filenames on top of FAT by having a UMSDOS file in each subdirectory holding the extra info.
- exFAT – advanced new FAT by Microsoft. Heavily patented so they can make money off of it. For use on SD cards. disks larger than 2TB, files larger than

4GB. No support for before windows XP, not backward compatible. Cluster size up to 32MB, many many other changes.