ECE 598 – Advanced Operating Systems
Lecture 14

Vince Weaver

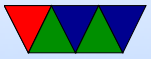
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

19 March 2015

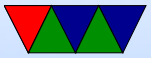
Announcements

- Homework #4 posted soon?

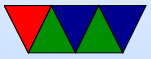


Filesystems

- Often a MBR (master boot record) and partition table
- Disks divided into paritions
  - Why partitions?
  - Split up system (/, /boot, /usr, /home)
  - Why is boot separate? Smaller so boot loader can access, maybe a different fs type.
  - Dual-booting operating systems
  - Swap partitions

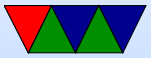


- Then individual filesystems

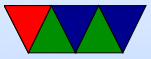


Filesystems – High Level

- Often first is called superblock, conaining all master info.
- Some sort of free list, saying what areas are free (bitmap or pointers)
- inodes, an array of data structures containing master info for each file (and if file is small, contents of file)
- root directory entry
- directory layout



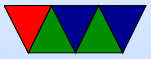
- file data



File Layout

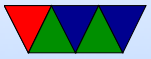
- Contiguous. Files consecutive blocks. Simple. Fast to read (just read X blocks) Has fragmentation problems like with memory alloc.
  Ever used? read-only, CD-ROMs

- Linked list. Inode points to first part, each block points to next. No fragmentation, seeking through file involves lots of reads.
  Can also instead have the pointers in one single block, each pointing to next block. File allocation table. Whole



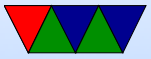
thing has to be in mem at once.

- inode table. List of blocks in file, last block reserved to point to next inode table.



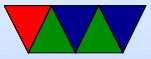
Disk performance

- Traditionally a lot of this came down to hardware.
- Spinning rust disks; head movement, cylinders/sectors. Reading consecutive faster, random access bad (milisecond bad)
  More complicated, fancy disk interfaces and embedded processors. Large caches (why can that be bad), shingled disks?
- Much of this goes away with flash disks, but still emulate



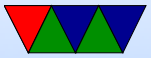
old disk interface

- Name lookup can also be slow.



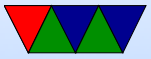
Common Filesystems

- Windows: NTFS, FAT
- Linux: EXT4, BTRFS
- OSX: HFS+
- Media: ISO9660, UDF



Fat FS

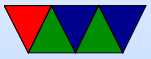
- FAT-12/FAT-16/FAT-32
- Various block sizes from 512 - 32kB (tradeoffs)
- Format
  - Boot Block – 512 bytes, first part configuration info (block size, blocks in disk, FATs, etc), rest actual boot loader code



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

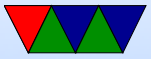
– One or more copies of File Allocation Table (FAT). Why multiple copies? Actually has to fit entirely in RAM.

  Just a table of 16-bit values, one for each cluster pointing to the next cluster in the file.



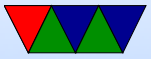
0	//////
1	//////
2	3
3	5
4	0
5	0
...	
N	0

– Root directory entry follows after last FAT.



8 bytes	filename
3 bytes	extension
1 byte	attr
10 bytes	reserved
2 bytes	creation time (h/m/s) second must be even
2 bytes	creation date
2 bytes	start cluster
4 bytes	filesize

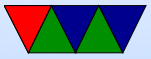
- Cluster size. Have to make it bigger to fit filesystems bigger than 32MB. Why can that be bad? (mostly, wasted space with small files)



- 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.

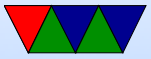
- FAT32 – increased sizes so can have max filsesize by 4GB.

- UMSDOS – linux filesystem that let you have



permissions, long filenames on top of FAT by having a UMSDOS file in each subrdirectory holding the extra info.

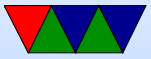
- exFAT – advanced new FAT by Microsoft. Heavily patented so they can make money off of it.



Ext2 FS

- All structures are little-endian (To aid in moving between machines)

- Format
  - Boot sector, boot block 1, boot block 2, boot block 3
  - Block group: superblock, fs descriptor, block bitmap, inode bitmap, inode table, data blocks
  - Superblock – located at offset 1024 bytes. Copies scattered throughout (fewer in later versions)

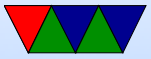


Info on all the inode groups, block groups, etc.

- Block descriptor table – description of how disk is split up into block groups
- Block bitmap – bitmap of blocks (1 used, 0 available)
- Inode bitmap – bitmap of available inodes
- Inode table – all metadata (except filename) for file stored in inode

  Second entry in inode table points to root directory
- Directory info – have an inode.

  Initial implementation was single linked list. Newer use hash or tree.



Holds inode, and name (up to 256 chars). inode 0 means unused.

- Hard links – multiple directory entries can point to same inode
- . and .. entries, point to inode of directory entry
- Subdirectory entries have name, and inode of directory