ECE 598 – Advanced Operating Systems
Lecture 18

Vince Weaver

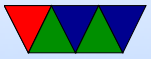
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

7 April 2015

Announcements

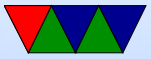
- HW#5 is graded and will be returned soon.
- HW#6 will be short answer, HW#7 will be graphics coding



HW#5 Review

1. File holes: write 4kb at beginning, seek 16MB write 4k
   Take up 16MB on disk? Or 8kb?
   Filesystem keeps track, only uses data blocks that aren't zero.
   Not the same as slack space. Slack space is when you have a file not an exact multiple of a block size. Zeros past the end, but there's not much you can do about that.

2. /proc/cpuinfo file

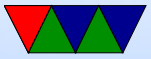


(a) Where is it stored? Not on disk.

A filesystem when registered with Linux provides a set of functions to the VFS that handle things like open, read, write, etc.

For a normal disk filesystem when you ask for data, the handler the filesystem registered will ask for blocks from the underlying block device and then pass them back to the program.

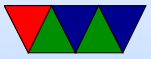
For a virtual program, the handler instead will create a block to return, but it is not read from disk. It is just



an area of memory and it can put anything it wants into it. In the case of /proc/cpuinfo, it notices the name of the file you want is "cpuinfo" and then just creates a disk block with the data you see. This is created on the fly, and in fact can change at any time (often due to CPU frequency scaling). It is created on the fly and so it doesn't really live anywhere, as the values are freed after you close the file.

(b) Why can you not remove a line?

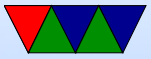
Most importantly, it might break the user/kernel ABI. If some program depended on that field being there



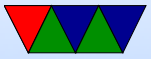
and crashes if it is removed, you just can't arbitrarily break the interface like that. You are stuck with bad ABI decisions forever.

3. Disk Quota – limits the amount of space a user or group can use.
   Most important on multi-user systems, or on fileservers.

4. Stateless – the server doesn't keep track of open files.
   Why is this useful? Server can reboot and clients can deal with this transparently. **Temporarily** go off line w/o noticing.

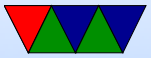


5. File Locking – only allows one file at a time to write.
   Why is this useful? Avoid files being corrupted.
   Do you need locking for reads? Read out of date info?
   Can you still corrupt files even with locking?



Graphics Interface History

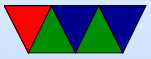
- Teletypes
- Vector Displays
- CRTs
- LCD displays



Video Display Technology

- Atari 2600 – Racing the Beam
  4k ROM, 128 bytes RAM, 40-pixel (5 byte) framebuffer
  3 sprites
  all calculation done during the retraces

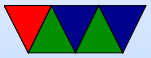
- SNES Tile/Sprite Based
  RAM getting cheap enough can have framebuffers, but bandwidth still not that great.
  Use tiles, that let you split the display into tiles, with each large tile specified by a single byte.



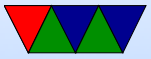
Video Adapters – Framebuffers

- Just an array of bytes that get displayed on the screen.

- Bits per pixel
  1 – monochrome
  4 – 16 colors
  8 – 256 colors (usually palette)
  15 – rgb 555
  16 – rgb 565 "true color"
  24 – rgb 888
  32 – rgba

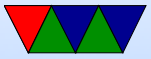


- Can be large: 1024x768x24bpp = 2.4MB, to update at 60Hz = 141MB/s
- Bit-planes
- Palette

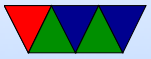


Video Adapters – GPUs

- Draw lots of triangles, really fast
- Can divide screen into small sections, and calculate massively parallel.
- OpenGL/Direct3d
- Triangles
- Textures

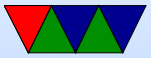


- Z-buffers
- Shaders
- Can make a card with no framebuffer? Text just written to texture and scaled to fill screen?



VGA Display example

- VGA text
- Memory mapped, IO ports
- Mode setting
- VESA BIOS
- “Mode X”
- Bitplanes



- Colors
- Loadable fonts